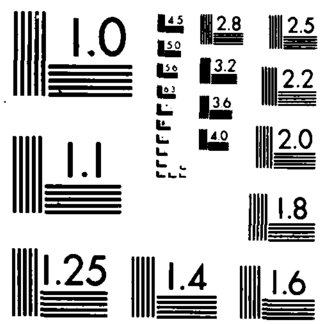
AD-A174 297 DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL 1/5
PACKAGE (ISTP) FOR LE..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF SYST.. K H CHUL
UNCLASSIFIED SEP 86 AFIT/GSM/ENC/86S-10 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL PACKAGE (ISTP) FOR LEARNING MATHEMATICAL CONCEPTS OF PROBABILITY AND STATISTICS

THESIS

Kim Hyung Chul
Captain, Republic of Korea Air Force

AFIT/GSM/ENC/86S-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

2

AFIT/GSM/ENC/86

DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL PACKAGE (ISTP) FOR LEARNING MATHEMATICAL CONCEPTS OF PROBABILITY AND STATISTICS

THESIS

Kim Hyung Chul
Captain, Republic of Korea Air Force

AFIT/GSM/ENC/86S-10

The contents of the document are technically accurate, and no sensitive items, detrimental ideas, or deleterious information are contained therein. Furthermore, the views expressed in the document are those of the author(s) and do not necessarily reflect the views of the School of Systems and Logistics, the Air University, the United States Air Force or the Department of Defense and the Republic of Korea Air Force and the Ministry of Defense.

Accession For		
NTIS GRA&I	☒	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A-1		

QUALITY INSPECTED 4

AFIT/GSM/ENC/86S-10

DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL PACKAGE (ISTP) FOR LEARNING MATHEMATICAL CONCEPTS OF PROBABILITY AND STATISTICS

THESIS

Presented to the Faculty of the School of Systems and Logistics
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Systems Management

Kim Hyung Chul, B.S.
Captain, Republic of Korea Air Force

September 1986

Preface

The purpose of this study was to develop an interactive statistical tutorial package, which is called the ISTP. This research was initiated to demonstrate an alternative pedagogy for teaching statistics. The ISTP was developed because traditional approaches to statistical education rely too heavily on using mathematics to motivate student comprehension of fundamental concepts.

In developing the ISTP and writing this thesis I have had a great deal of help from my thesis advisor Dan Reynolds. A note of thanks and appreciation is extended to Rich Lamb for his mathematical advice and consultation throughout the period of developing the ISTP and writing of this thesis. Without such help this research would not have been possible. Any questions regarding the ISTP should be addressed to Professor Dan Reynolds, AFIT/ENC, AV 785-4185.

Finally, special love and appreciation are offered to my wife, Hyeja, and my daughter, Minyung, for their deep understanding and assistance through the countless days and nights of this research and AFIT program.

Table of Contents

List of Figures

List of Tables

Table | Page

AFIT/GSM/ENC/86S-10

Abstract

This ~~research~~ thesis demonstrated the feasibility of an alternative pedagogy offered by the ISTP: one that emphasizes visual presentation and mastery of statistical concepts in lieu of the traditional theorem-proof approach to learning statistics.

Four statistical topics were selected for the initial implementation of the ISTP: probability distributions, maximum likelihood estimation, transformation of random variables and hypothesis testing. These areas were chosen because they are the main concern of a first course in statistics.

The ISTP was designed to provide an intuitive pedagogy rather than to present statistics in a cook-book fashion. The user is encouraged to think while using the ISTP and to relate the results of the ISTP to concepts of statistics.

The ISTP is hierarchically constructed so that the beginner can use this package by taking advantage of menu options. For the experienced user, it provides direct macro selection. Furthermore, the more sophisticated user can develop his own library of macros to add to those which already exist in the ISTP.

Further research should attempt to enhance current subject matter and take advantage of graphics facilities

such as color graphics, animation, and windowing. More specifically, research needs to be conducted on subjects covered by current topic areas as well as more advanced subjects such as axiomatic probability, Bayesian statistics, and multivariate probability distributions. Finally, consideration should be given to porting the entire ISTP package to super computers to cope with the computational needs of forthcoming graphic packages.

DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL PACKAGE (ISTP) FOR LEARNING MATHEMATICAL CONCEPTS OF PROBABILITY AND STATISTICS

I. Introduction

General Issue

Traditionally, statistical pedagogy has promoted the use of the formal language of mathematics to teach concepts of probability and statistics. For most students, this involves simultaneous mastery of a foreign language (mathematics) and of abstract notions of probability and statistics. This research effort was initiated to demonstrate the feasibility of an alternative pedagogy: one that emphasizes rigorous visual presentation and mastery of statistical concepts in lieu of the traditional theorem-proof approach to learning statistics. It was assumed that such an approach would encourage early and sustained motivation for the subject matter of statistics and foster competent application of statistical technologies during student research efforts.

Problem Statement

Air Force Institute of Technology (AFIT) graduate students need to have statistical analytic capabilities because their thesis research often requires statistical

analyses. However, most students are unfamiliar with statistics when they start their graduate programs. The traditional lecture/lab approach to teaching statistics needs to be supplemented because, in class, it is difficult for instructors to explain statistical theory without using complicated mathematical theories or formulae. In general, most students cannot cope with such complicated mathematical presentations. Fortunately, the computer can pictorially present statistical theory so that students can understand the concepts of visualization rather than by derivation of mathematical theorems or algorithms. Current statistical computer packages usually facilitate numerical computation but offer little assistance to users in understanding concepts of statistics. Therefore, a graphics-based interactive statistical computer package needs to be developed to provide such tutorial assistance for fundamental areas of statistics.

Research Objectives

The goal of this research was to develop a prototypical graphics-based interactive statistical package that could be used by students in statistical courses taught at AFIT. It was assumed that the Interactive Statistical Tutorial Package (ISTP) could provide non-mathematical explanations of statistical concepts by utilizing computer graphics and thus shield students from heavy involvement with the mathematical detail. To achieve

the research goal, two research objectives were established: (1) to select four topics from the domain of mathematical statistics which need to be mastered by all graduate students taking statistics courses at AFIT, and (2) to develop an ISTP that could demonstrate the feasibility of an alternative pedagogy by taking advantage of visual presentations without heavy involvement with mathematics.

Scope of the Research

The study focused on making an interactive statistical tutorial package ISTP. No formal attempt was made to evaluate the effect of using the ISTP.

The ISTP was designed to address four subject areas of statistics. Several prototypical learning modules were developed under each of these subject areas.

The interactive statistical computer language S was used as the primary language for developing software for the ISTP because it is currently available to all students at AFIT. Furthermore, compared to other interactive statistical languages, S has many strengths that make it easy to learn and easy to use. For example, it has a macro-capability that allows users to develop their own libraries and excellent graphics.

The ISTP was developed within the environment of S provided by AFIT computer systems without regard to its portability to other environments.

Definition of Terms

1. AFIT Graduate Students. AFIT graduate students are those who are enrolled in the Air Force Institute of Technology School of Systems and Logistics or the Operations Research program at the AFIT Engineering School. Typically, they are required to take two or three probability and statistics courses as a part of their program and usually employ statistical technology during their thesis research.

2. ISTP: The Interactive Statistical Tutorial Package. The ISTP is a set of prototypical learning modules developed to demonstrate the feasibility of an alternative pedagogy for learning statistics by using a visually-oriented approach in lieu of the traditional theorem-proof approach of teaching statistics.

ISTP communicates with users. It requires input from the user and gives output to the user interactively and dynamically based upon changing inputs. Communication between the user and the computer is accomplished by menus and printed messages.

3. S: An Interactive Statistical Computer Language. "S is an interactive environment for data analysis and graphics" (5:preface,i). S is designed to provide statistical computation, data management, and graphics output. As an interactive language, it encourages real time communication between user and computer and thus can provide each user with efficient feedback.

Statistical packages such as SPSS-x, BMDP, or SAS require users to predetermine what procedure needs to be run prior to execution of the program. S, on the other hand, allows users to explore the nature of data and to create a procedure concurrently with data analysis.

Assumptions

The first assumption is that the need for statistical education for AFIT students' research will continue.

The second assumption is that not only statistical packages such as SPSS-x, BMDP, or SAS, but also interactive statistical languages such as S, will be used in statistics classes.

The third assumption is that AFIT graduate students have a limited background in statistics and generally little experience with mathematics beyond core courses in calculus when they start their program.

The fourth assumption is that human beings prefer to take a visually-oriented approach to learning statistics rather than to take a rigorous analytic approach in understanding statistics, because the visually-oriented approach can provide concrete images or visualization of abstract statistical concepts.

Background

AFIT graduate students have serious time constraints in their graduate programs. They have only five

or six academic quarters and are required to take more than sixty credit hours, including completion of a master's thesis.

Many AFIT graduate students conduct their research by surveys or experiments. Such procedures require statistical analyses. That is why AFIT graduate students have to take two or three consecutive probability and statistics courses. However, in some cases, graduate students cannot get enough capability to apply statistics to their research properly. Too much time is spent on learning the mathematical theory of probability. Too little time is spent on applying statistics to real-world problems. The fact that this is not only a problem for AFIT students is confirmed by Professor Welsch of the Massachusetts Institute of Technology, who stated, "most users of packaged programs are likely to have only one course in statistics, and that one often has a third of the time spent on probability" (19:34).

The time constraint on statistics courses is a serious problem, particularly when professors of statistics choose to use mathematical theories or formulae to explain the rationale for various statistics. This forces students to review formal mathematics concurrently with their attempt to learn the language of statistics. Most students find this approach to teaching statistics counterproductive.

Clearly, tradeoffs must be considered between teaching applied statistics and the mathematical theory of statistics. Neither can be ignored. Using the theorem-proof approach usually precludes adequate coverage of applied statistical material. On the other hand, too much emphasis on a cook-book approach fosters mindless use of statistical methods. At this point, the need for an alternative pedagogy for learning statistics should be obvious. The ISTP provides such an alternative pedagogy by using a graphics-based approach to prompt the understanding of statistical concepts without requiring students to involve themselves in a mathematical theorem-proof approach. This expedition of understanding statistical concepts not only provides students with positive motivation for studying statistics but also allows students time to apply statistics to real-world problems and to learn how to intelligently use statistical packages such as SPSS-x, BMDP, and SAS.

When the need for a new approach to learning statistics became apparent, an effort was made (1) to review germane research concerned with using computers for statistical education, and (2) to determine whether any tutorial packages had yet been developed. Although two interactive statistical packages were found, neither could adequately meet all the elements of the stated need.

Consequently, an effort was mounted to develop a new statistical package named the ISTP to fulfill the objectives of the research.

II. Literature Review and Research Methodology

Literature Review

Using Computer Packages for Statistics. Many instructors recognize that statistical computer packages are essential for statistics courses and most of them actually use such computer packages in statistics classes. Professor Thisted of the University of Chicago recommends that computing based on popular statistical software should be taught in the second course of statistics in a two-course sequence. His main ideas about teaching statistical computation can be summarized as follows:

> 1. Several packages must be studied so that students learn to distinguish features of statistical software in general from peculiarities unique to an individual product. . . .
> 2. Several large data sets should be examined using the packages. . . .
> 3. Computational issues affect statistical practice; the outcome of a research project may depend critically on the efficiency and numerical accuracy of algorithms employed and on the manner in which a large data set is physically organized.
> 4. There are statistical issues unique to the analysis of large problems on computers. (18:27-28)

Professor DuMouchel, a professor of Applied Mathematics at the Massachusetts Institute of Technology, commenting on Professor Thisted's paper in The American Statistician, states that statistical concepts should be understood before statistical computation is introduced, because comparative discussions of statistical software

will be beyond a student's comprehension who is still struggling with fundamental concepts like dispersion, sampling, significance, and so forth (10:30). Furthermore, Professor DuMouchel suggests that computer packages, especially interactive ones, can be very useful in teaching a first course in statistics (10:30). He provides the following example:

> It is important, in my opinion, to include interactive computer use in the very first data analysis course, if only to put our best foot forward and show that statistics can be relevant, useful, and interesting (even Fun!). Many students have told me that although they had taken one course in statistics, it did more harm than good because it confused and frightened them and they just wanted to start over. Computer packages, especially interactive ones, can help the students avoid excessive focus on the mathematical procedures, manipulations, and calculations required, and concentrate on the intuitive, common sense basis of the various inferential techniques and on the assumptions behind them. (10:30)

After reviewing Professor DuMouchel's comments concerning his original paper, Professor Thisted agreed that interactive packages are very useful, especially for beginners, because "interactive computing is participatory and dynamic; the student must think, react, and interact. What is more, feedback is immediate" (18:35).

The above comments suggest that a tradeoff must be made concerning the use of statistical packages. If one waits until the second course before introducing the statistical packages, very little time remains for dealing with problems of applying statistics to the real world. On the other hand, if one chooses to exploit computer

technology during the first course concurrent with discussions about probability, theoretical notions may be treated in such a superficial manner that the student will be encouraged to approach statistics in a cook-book fashion. This appears to pose an insoluble dilemma; theory and applications both need to be covered, but cannot be addressed adequately using traditional pedagogy. Clearly, an alternative pedagogy needs to be proposed if the dilemma is to be resolved. Therefore, a search was made to see if statistical software existed that could support such an alternative.

Interactive Statistical Computer Package. During the literature review for this thesis two packages surfaced: (1) ISP: Interactive Statistical Programs, and (2) CLT: Central Limit Theorem program. ISP is "a comprehensive system for learning and teaching purposes" (14:140).

> It is specially designed to teach a basic statistics course to students in social science and management who have no quantitative background and no experience with computers. It enables teachers and students to concentrate on statistical concepts rather than on statistical computations. Moreover, provisions for working with real data, sampling from data sets, conducting simulation, and experimenting with games of chance help to make statistics more interesting and give students a better understanding of the nature of uncertainty, sampling, variability, and statistical modeling. (14:140)

It offers a user-friendly introduction to the computational power of statistical packages.

CLT is a tutorial-type package that provides an interactive approach to learning the central limit theorem (13:90). The purpose of the CLT program is to illustrate the notion of a limiting distribution proposed by the central limit theorem. It can graphically demonstrate that the sampling distribution of the mean approaches normality as the size of the sample increases. This efofort to motivate a statistical concept by a graphics-based approach assumes that people, including persons who have a strong mathematical background, can understand a statistical concept better by visual confirmation of its legitimacy than by a formal mathematical proof of its truthfulness.

The weakness of the CLT program is that it addresses only one of many statistical notions. The problem with ISP is that, while it encompasses the full breadth of elementary statistical theory, its presumption of statistical naivete on the part of users severely limits its pedagogical applicability. To provide adequate coverage for all levels of pre-statistical mathematical competency, a new package had to be developed. Since this package would take a graphics-based approach to selected areas of mathematical statistics, the next part of the literature search involved finding justification for the use of graphics in the learning process.

Application of Graphics. A great deal of research has been accomplished in the area of graphics to prove its

power over verbal presentation or presentation of raw data. An article by Maivald captures the essence of the position taken by this research. Maivald talks about the role the computer can play in the generation of graphics images.

> Using computers as image generators complements the physiological properties of the brain through the computer's offering of high speed, library access, and its ability to show, in color, the dynamics of the observed phenomenon. (15:84)

She then presents an example from integral calculus, normally addressed in mathematical terms, as it might be taught using a graphics approach. She defines velocity as in Equation 1 (15:83):

$$V = \int_{T(o)}^{T} A(T) \cdot dT \qquad \text{or} \qquad V = \lim_{\Delta T \to o} \sum_{T(o)}^{T} A(T) \cdot \Delta T \qquad (1)$$

where A is acceleration and T is time.

Next, she interprets velocity graphically as in Figure 1 (15:83).

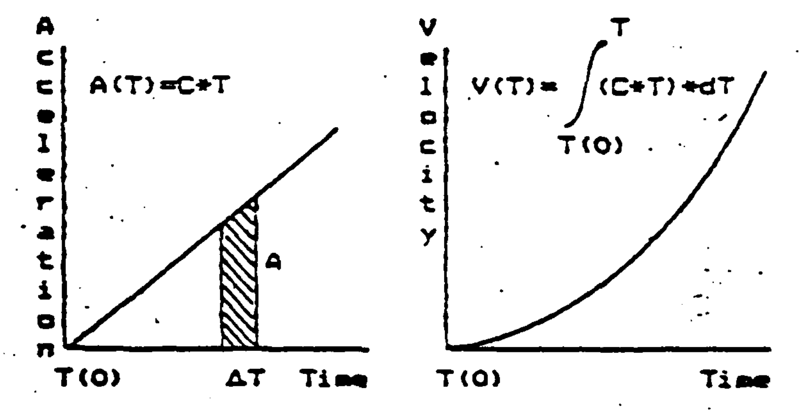


Fig. 1. Graphical Interpretation of Velocity

As shown in Figure 1, "the simulated summation in graphical mode is observed as a gradually shaded area with a tally running simultaneously" (15:83). Maivald points out that the computer is the best tool to display dynamic phenomenon:

> Computers have a large structured memory and the ability to display graphical data in a dynamic condition and thus can help with the image component of complex code. They are far superior to human teachers in this area. (15:84)

Although Maivald presents a cogent argument for introducing graphics presentations into the learning process, it is not necessarily true that graphics presentations are better in every case. Graphics can provide users with trends of the data or the general ideas of verbal propositions but cannot facilitate presentation of exact data points or precise mathematical definitions. In most cases, it complements non-graphical modes of information display.

Therefore, this research emphasizes the use of the computer as an image generator for the new statistical package. One of the reasons S was selected as the primary language for the ISTP is that S has many graphics functions. In the following section, S's unique capability for such graphical imaging and other types of data analysis are reviewed.

S Interactive Statistical Language. As stated before, "S is an interactive environment for data analysis

and graphics" (5:preface,i). Since S provides the user with interactive computation, both simple and complex, graphic displays on a wide variety of graphics devices and many alternative ways to manage data and data structures, S is quite unlike most statistical packages among the many statistical packages or languages (5:preface,i). The interactive computational function of S shields users from complicated mathematical computations. The S graphics functions provide pictographic examples of applied statistical concepts for users.

For the following reasons, S was selected as the primary language to develop the ISTP software.

1. S has many statistical functions. There are ten continuous type distributions for which S can directly calculate probabilities and quantiles. It can also generate random deviates from these distributions (3:6). Probabilities for discrete distributions such as the Poisson or Binomial can be computed by S because of the relationship between the binomial distribution and the beta distribution and between the Poisson distribution and the gamma distribution (3:7). S also has statistical functions for computing measures such as the mean or variance of sample values.

2. S has a graphics capability. S can provide histograms, two- and three-dimensional plots, and contour plots. Inexperienced users can use the graphic functions

easily by invoking default modes. Experienced users can obtain desired graphs by giving proper arguments instead of default arguments.

3. S has a macro capability. The S macro facility "allows new function-like operations to be built up from S expressions, and allows commonly used expressions to be saved and executed with a minimum of typing" (5:155).

4. S allows the user to create new functions. S can invoke existing algorithms in FORTRAN through an interface routine. Because it is inefficient to write a macro when the computation involves iterating over single elements of the data, creating new functions which can use external libraries such as IMSL in FORTRAN, is a better approach than creating macros (4:4).

5. S facilitates guided access to program modules. S allows messages to be contained in macros as well as menu-driven interfaces. This menu capability permits users to use developed packages hierarchically. By selecting one or several possible lower-level menus, the user can reach the level he wants (5:172).

Conclusion. Statistics plays a very important role in any AFIT graduate student's thesis research. Before the statistical computer packages such as SPSS-x, BMDP, or SAS are used, statistical concepts related to their use should be studied. However, current interactive statistical

packages only partially meet the need for a tutorial statistical package and dynamic presentation.

Research Methodology

The goal of this research was to develop a prototypical graphics-based interactive statistical package. To achieve this research goal, two research objectives were set up: (1) to select subjects from the domain of mathematical statistics, and (2) to develop the ISTP: Interactive Statistical Tutorial Package. In order to meet these objectives, the following methodology was employed.

Selection of Statistics Subjects. Four areas of statistics were selected. These selected subjects were probability distributions, maximum likelihood estimation, transformation of random variables, and selected topics of hypothesis testing. Clearly, there are more statistical topics than the selected ones. However, the four selected topics are the main concern of the first course in mathematical statistics. The justifications for selecting these subject areas will now be outlined.

Probability Distributions. Probability distribution models provide the foundation for all forms of statistical estimation and testing. Understanding of probability distributions in terms of probability mass functions (PMFs) or probability density functions (PDFs)

is fundamental to the study of probability distributions in general. Generating random variates is important to Monte Carlo simulation and goodness of fit testing.

Maximum Likelihood Estimation. The maximum likelihood methodology is the most popular means for obtaining point estimates. "The objective of point estimation is to compute from the sample a single number which will be our best guess for the true value of the parameter under investigation" (9:209).

Point estimation is an important topic in the study of probability distributions. Because of its importance, maximum likelihood estimation was selected as the second topic of the ISTP.

Transformation of Random Variables. The third topic selected for implementation for the ISTP was transformation of random variables. It is necessary to study such a technique in order to appreciate the derivation of test statistics which are, in fact, functions of random variables. Also, because the relationship between distributions such as normal and log-normal can be understood as the transformation of a random variable, this area was considered essential to the initial implementation of the ISTP.

Hypothesis Testing. Hypothesis testing was the final area selected because it plays a central role in all student research.

Developing Statistics Package. For each of the four subject areas, ISTP routines were developed to visualize and dynamically display the concepts embodied by the topical areas. The essential areas of statistical theory encompassed by the learning modules of the ISTP addressing each subject area are enumerated below.

Elements of Probability Distributions. For most univariate distributions and for some bivariate distributions, an attempt was made to compute probabilities and quantile values, generate random variates, and display the shapes of distributions by using PDFs (PMFs) or cumulative density functions (CDFs).

S has functions which compute the probabilities and quantiles and generate the random variates for most univariate continuous distributions. But there are no functions in S to compute probabilities or generate random variates for the univariate discrete distributions or bivariate continuous distributions. Although probabilities for some univariate discrete distributions can be computed due to the relationship to the univariate continuous distributions, S macros were developed which could compute the probabilities and generate random variates for the univariate discrete distributions and bivariate continuous distributions. This expanded existing S functions so as to provide a complete statistical package.

Most univariate distributions, including both discrete and continuous, and some bivariate distributions, were studied in this research in order to compute their PDFs (PMFs) and CDFs, generate random variates, and plot the shape of each distribution. However, multivariate distributions were considered to be beyond the scope of this research.

Elements of Maximum Likelihood Estimation. Estimation of population parameters using the maximum likelihood method is attractive when sample sizes are large. However, in this research, sample sizes of one or two were studied because this package focuses on the concept of maximum likelihood estimation rather than the actual computation of parameter estimates. Using sample sizes of one or two, computer graphics can provide meaningful displays of maximum likelihood estimation in two or three dimensions.

The graphics approach to maximum likelihood estimation allows the user to input reasonable guesses which substitute for parameter values that might maximize the joint PMF or PDF for the given sample and then provide the maximum likelihood estimation for the parameters so that the user can have an idea how good his guesses were.

Elements of Transformation of Random Variables. Random variables are transformed using a graphics method that minimizes the user's exposure to

calculus based on techniques such as the Jacobian method or method of moments.

In this research, instead of integrating the PDF to obtain the CDF of univariate random variables, the ISTP uses existing S functions or specially developed macros. Double integration routines of the IMSL library were executed to obtain the CDF of bivariate random variables.

Elements of Hypothesis Testing. Hypothesis testing was selected as a final topic of the ISTP. Population distributions, sampling distributions, test statistics, likelihood ratio statistics, the power of tests and the significance of the test results are displayed in graphic form.

From among the many hypothesis test procedures, tests about the mean of a normal distribution when the standard deviation (σ) is known and unknown were selected for implementation in the ISTP.

Overview of the Following Chapters

In Chapter III, the concepts of the four selected areas and pedagogical rationale used to develop the ISTP are introduced. Examples and procedures for using the ISTP are presented in Chapter IV in order to provide a guide and the necessary documentation for using the ISTP. In Chapter V, conclusions about lessons learned during the development of the ISTP are drawn and recommendations are

made concerning future research and extensions that might be made to the current version of the ISTP.

III. Rationale for the Pedagogy Implemented by the ISTP

The ISTP was developed to fulfill the need for a graphics-based tutorial computer package which could foster understanding of statistical concepts through a visually-oriented approach appealing to the student's intuition rather than by explanations which presume competency with the core elements of calculus. Thoughout this chapter, the concepts of the four statistical areas covered by the ISTP and the pedagogical rationale used to develop the tutorial software for these concepts are introduced. The material of this chapter lays the foundation for presenting specific applications of the ISTP in Chapter IV.

The chapter is arranged into four sections, dealing with probability distributions, maximum likelihood estimation, transformation of random variables, and hypothesis testing. Each section focuses on one of the four topics and discusses the motivation for constructing pedagody for teaching the underlying concepts associated with that topic. The format of procedures used to develop the learning modules for each topical element is then outlined. This explication is typically clarified by an accompanying illustrative example.

Probability Distributions

The ISTP package extends S's capability to calculate probabilities and quantiles of random variables by offering a complete menu of calculations for univariate discrete and continuous random variables and selected bivariate discrete and continuous random variables. The ISTP was invested with such computational power to free the student from burdensome and tedious use of tables in text books affording him the opportunity to explore the dynamics of alternative probability computations. The software of the ISTP can be asked to graphically display most of the above probability distributions. The package permits the user to select parameter values for plotting probability distributions so the user is encouraged to attain a panoramic view of family membership of any particular probability distribution. Random samples of size n from both univariate and bivariate distributions can be produced using the stochastic generators of the ISTP. This facilitates users' simulation and analysis of data that arise in the real world.

Univariate Continuous Distributions. Although S has functions for ten continuous type distributions, there are no explicit functions for three well known and frequently used continuous distributions: the Exponential, the Weibull and the Triangular distribution. The exponential distribution can be considered as "a special case

of the general gamma PDF in which $\alpha = 1$ and β has been replaced by $\frac{1}{\lambda}$" (9:152). However, since "there are gamma distributions which are not Weibull distributions and vice versa," (9:157) new functions had to be added to S to calculate the PDF, the CDF and quantiles and to generate random variates for the Weibull distribution. Table 1 lists functions in S and macros in the ISTP which compute PDFs, CDFs and quantiles and generate random variates for univariate continuous distributions.

Computation of the PDF and the CDF. As long as the PDF of a random variable exists and the CDF can be expressed in closed form, the PDF and the CDF can be obtained by calculation using actual parameters and/or quantile values. For example, although S provides a function dgamma to compute the PDF of the gamma random variable with one parameter, the so-called standard gamma, it is often necessary to obtain the PDF of the gamma random variable with two parameters displayed as Equation (2) (9:150). Therefore, the macro ?gamfun was built into the ISTP to compute the density for this two-parameter gamma random variable.

$$f(x;\alpha,\beta) = \begin{cases} \frac{1}{\beta^{\alpha}\Gamma(\alpha)} \cdot x^{\alpha-1} e^{-x/\beta} & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where $\alpha > 0$, $\beta > 0$.

TABLE 1

FUNCTIONS IN S AND MACROS IN THE ISTP FOR UNIVARIATE CONTINUOUS DISTRIBUTIONS

Type of Distributions	PDF		CDF		Quantile		Random Variate	
	S	ISTP	S	ISTP	S	ISTP	S	ISTP
Normal	dnorm		pnorm		qnorm		rnorm	
Log-Normal	dlnorm		plnorm		qlnorm		rlnorm	
Uniform	dunif		punif		qunif		runif	
T	dt		pt		qt		rt	
F	df		pf		qf		rf	
Chisquare	dchisq		pchisq		qchisq		rchisq	
Standard Gamma	dgamma		pgamma		qgamma		rgamma	
Gamma(2 para)		?gamfun						
Beta	dbeta		pbeta		qbeta		rbeta	
Cauchy	dcauchy		pcauchy		qcauchy		rcauchy	
Logistic	dlogis		plogis		qlogis		rlogis	
Exponential		?dex		?pex		?qex		?rex
Weibull		?dweib		?pweib		?qweib		?rweib
Triangular		?dtrian		?ptrian		?qtrian		?rtrian

<u>Generating</u> <u>Random</u> <u>Variates</u>. Quantile values can be computed and random variates can be generated by using the inverse transform technique (1:294-300). An algorithm for the inverse transform technique, illustrated using the exponential distribution, follows (1:294-295):

Step 1. Compute the CDF of the desired random variable X. For the exponential distribution, the CDF is $F(x) = 1-e^{-\lambda \cdot x}$, $x \geq 0$.

Step 2. Set $F(X) = R$ on the range of X. For the exponential distribution, it becomes $1-e^{-\lambda X} = R$ on the range $x \geq 0$.

Since X is a random variable (with the exponential distribution in this case), it follows that $1-e^{-\lambda X}$ is also a random variable, here called R. As will be shown later, R has a uniform distribution over the interval (0,1).

Step 3. *Solve the equation F(X) = R for X in terms* of R. For the exponential distribution, the solution proceeds as follows:

$$1 - e^{-\lambda X} = R$$

$$e^{-\lambda X} = 1 - R$$

$$-\lambda X = \ln (1-R)$$

$$X = \frac{-1}{\lambda} \ln (1-R) \tag{3}$$

Equation 3 is called a random variate generator for the exponential distribution. In general, equation is written as $X = F^{-1}(R)$.

Step 4. Generate (as needed) uniform random numbers R1, R2, R3, ... and compute the desired random variates by

$$X_i = F^{-1}(R_i)$$

For the exponential case,
$F^{-1}(R) = (-1/\lambda)\ \ln(1-R)$ by equation 3, so that

$$X_i = \frac{-1}{\lambda} \ln(1-R_i)$$

The same technique was applied to generate Weibull deviates and triangular deviates and to compute quantile values for their respective cumulative probabilities.

<u>Univariate Discrete Distributions</u>. Although S does not have any function for discrete distributions, PMFs for some discrete distributions are computable due to their relationship to other distributions in S. Relationships between the binomial distribution and the beta distribution and between the Poisson distribution and the gamma distribution enable the computation of discrete probabilities for the binomial distribution and the Poisson distribution (3:7).

The relationships between the binomial distribution and the beta distribution and between the Poisson distribution and the gamma distribution are as follows (2:1).

If $X \sim$ binomial (n, p) and $Y \sim \mathrm{Beta}(x+1, n-x)$, then

$P(X \leq x) = P(Y \geq p)$. Therefore,

$P(X \leq x) = 1 - \mathrm{pbeta}(p, x+1, n-x)$

$P(X \geq x) = 1 - P(Y \geq p) = P(Y \leq p)$, where now

$Y \sim \mathrm{Beta}(x, n-x+1)$.

Therefore,

$P(X \geq x) = \text{pbeta}(p,x,n-x+1)$

$P(X=x) = P(X \leq x) + P(X \geq x) - 1$. Therefore,

$P(X=x) = \text{pbeta}(p,x,n-x+1) - \text{pbeta}(p,x+1,n-x)$.

For example, let X~binomial(16, 0.45), then

$P(X \geq 10)$ can be found in S via

```
> pbeta(0.45, 10, 7)
  0.124103
```

And P(X=10) can be found in S via

```
> pbeta(0.45, 10, 7) - pbeta(0.45, 11, 6)
  0.0754788
```

And if X~Poisson(m), and Y~gamma(x, 1), then

$P(X \geq x) = P(Y \leq m)$. Therefore,

$P(X \geq x) = \text{pgamma}(m,x)$

$P(X \leq x) = 1 - P(Y \leq m)$. Therefore, where now

Y~gamma(x+1,1)

$P(X \leq x) = 1 - \text{pgamma}(m,x+1)$

$P(X=x) = P(Y \geq m) + P(Y \leq m) - 1$. Therefore,

$P(X=x) = \text{pgamma}(m,x) - \text{pgamma}(m,x+1)$

For example, let X~Poisson(4.5), then

$P(X \leq 6)$ can be found in S via

```
> 1 - pgamma(4.5, 7)
  0.83105
```

P(X=6) can be found in S via

```
> pgamma(4.5, 6) - pgamma(4.5, 7)
  0.1281199
```

Table 2 lists available macros in the ISTP that compute PMFs and CDFs and generate random variates for the univariate discrete distributions.

Computation of the PMF. Most PMFs for discrete distributions exist. For example, the PMF for the binomial distribution is (9:95):

$$b(x;n,p) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & x = 0,1,2,\ldots, n \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Equation (4) for x = 0, 1, 2, ..., n can be developed as follows, when b(x;n,p) = Y;

$$Y = \frac{n!}{x!(n-x)!} \cdot p^x \cdot (1-p)^{n-x}$$

$$= \frac{\text{gamma}(n+1)}{\text{gamma}(x+1) \cdot \text{gamma}(n-x+1)} \cdot p^x \cdot (1-p)^{n-x} \tag{5}$$

Equation (5) can be replaced by Equation (6). The reason the natural logarithm is taken on both sides is that by doing so S can compute the combination when n is large. Any n >= 34 will cause gamma(n+1) to produce an overflow condition.

$$\ln(Y) = \text{lgamma}(n+1) - \text{lgamma}(x+1) - \text{lgamma}(n-x+1) + x \cdot \ln(p) + (n-x)\ln(1-p) \tag{6}$$

TABLE 2

MACROS IN THE ISTP FOR UNIVARIATE DISCRETE DISTRIBUTIONS

Type of Distributions	PMF	CDF	Random Variate
Bernoulli	?mber		?rber
Binomial	?mbin	?bin	?rbin
Negative Binomial	?mnegbin	?negbin	?rnegbin
Geometric	?mgeo	?geo	?rgeo
Hypergeometric	?mhyper	?hyper	?rhyper
Uniform	?ddunif	?pdunif	?rdunif
Poisson	?mpoiss	?poiss	?rpoiss
ReyLam	?mreylam		?rreylam

Equation (6) can be solved in terms of Y to obtain the PMF for the binomial distribution;

$$Y = \exp\left\{ \mathrm{lgamma}(n+1) - \mathrm{lgamma}(x+1) - \mathrm{lgamma}(n-x+1) + x \cdot \ln(p) + (n-x) \cdot \ln(1-p) \right\} \qquad (7)$$

Equation (7) is used in macro <u>?mbin</u> to obtain the PMF for binomial distribution. PMFs for other discrete distributions, such as the negative binomial, the hypergeometric, the uniform, the Poisson, the Bernoulli, the geometric, and the ReyLam distribution were obtained using a technique similar to that used for the calculation of the binomial distribution.

The ReyLam distribution is essentially a point multinomial in which random variables can attain three values and the sample size is one. For n = 1, the ReyLam distribution is defined with following PMF:

$$p(x) = \begin{cases} p_1 & x = 0 \\ p_2 & x = 1 \\ 1-p_1-p_2 & x = 2 \\ 0 & \text{otherwise} \end{cases}$$

As demonstrated below, the computation for the PMF of the binomial distribution and the Poisson distribution from their macros provides the same results as previously obtained using the beta distribution and gamma distribution.

```
> ?mbin(10, 16, 0.45)
  0.075479
```

gives the same result as using pbeta(.45, 10, 7) -pgamma(.45, 11, 6).

```
> ?mpoiss(6, 4.5)
  0.1281201
```

gives the same result as using pgamma(4.5, 6)-pgamma(4.5, 7).

Computation of the CDF. The CDF for a discrete distribution is calculated by summing up their PMFs as shown in the following example with the binomial distribution (9:95):

For $X \sim Bin(n,p)$, the cumulative distribution function is denoted by

$$P(X \leq x) = B(x;n,p) = \sum_{y=0}^{x} b(y;n,p), \quad x=0,1,\ldots,n$$

Therefore, once the PMF for a discrete distribution is defined, the CDF can be obtained by summing up its PMF for the desired X. For example, in the ISTP, the PMF function for a binomial distribution is defined as macro ?mbin(x,n,p), where x is the value of the random variable, n is the number of trials, and p is the probability of success, so that the following method can be used to obtain the CDF for the binomial distribution:

$$P(x_1 \leq X \leq x_2) = \text{sum}(?\text{mbin}(x_1:x_2,n,p))$$

Another macro that computes the CDF is defined in the ISTP; namely, ?bin(x_1,x_2,n,p).

The computation for the CDF of the binomial and Poisson distributions provides the same results as previously obtained using the beta distribution and gamma distribution. For example,

```
> ?bin(10, 16, 16, 0.45)
  0.1241033
```

gives the same result as using pbeta(0.45, 10, 7), and

```
> ?poiss(0, 6, 4.5)
  0.83105
```

gives the same results as using 1-pgamma(4.5, 7).

<u>Generating Random Variates</u>. The inverse transform technique, which was used to generate random variates for continuous distributions, can also be applied to generate random variates for discrete distributions. Recall that the CDFs of discrete random variables are step functions, while the CDFs of continuous distributions are continuous functions. When the inverse transform technique is applied to generate random variates for discrete distributions, a set of random numbers (R) which fall into the range of $F(X_{i-1}) < R \leq F(X_i)$ should be mapped to X_i as the value of the random variate. This technique was used to generate random variates for the Bernoulli, the uniform, the hypergeometric and the ReyLam distribution.

Binomial random variables are computed by summing n Bernoulli random variables.

Since values of the random variables for the geometric distribution, the negative binomial distribution,

and the Poisson distribution range from 0 to infinity, exact PMFs and CDFs are required to obtain the accurate random variates. This involves making a decision regarding the level of required accuracy.

For the geometric distribution, PMFs for random variables from $x = 0$ to $x = \frac{5}{p}$, where p is a parameter of probability of success for the geometric distribution, give sufficient accuracy, because the CDF for $0 \leq x \leq \frac{5}{p}$ is approximately 1 except for the case of very small p. In this case, PMFs are computed from 0 to 30 instead of from 0 to $\frac{5}{p}$. For the same reason, PMFs for the Poisson random variable are computed from 0 to 5λ, where λ is the Poisson parameter, When λ is small, 30 can replace 5λ.

PMFs for the negative binomial random variables are computed by applying the binomial probability function. The probability that R successes will occur given R+m trials for which each trial having probability of success equals p is a binomial probability. m is the number of failures prior to the Rth success and is a negative binomial random variable. Therefore, each PMF for $m = 0$ to $m = \infty$ can be computed, and the sum of PMFs of the binomial random variables for $m = 0$ to $m = \infty$ is $\frac{1}{p}$, where p is the probability of success for each trial. And the sum of PMFs for $m = 0$ to $m = 5 \cdot \frac{R}{p}$ approaches $\frac{1}{p}$ except for small m, so that $5 \cdot \frac{R}{p}$ is replaced by 30 for the case of small m. Since the maximum of the CDF accumulates to $\frac{1}{p}$, multiplying

p over the CDFs will make the range of the CDF 0 to 1. Then the inverse transform technique can be applied to generate random variates for the negative binomial distribution.

Bivariate Continuous Distributions. If two random variables X and Y are independent, the bivariate PDF is the product of their separate PDFs. However, this is not the case when they are dependent, and a very useful example of such a correlated model is the bivariate normal distribution with correlation not equal 0. In this research, the bivariate normal distributions for dependent and independent random variables and the bivariate gamma distribution for independent random variables were studied to explore the properties and dynamics of bivariate distributions. Table 3 lists available macros which compute PDFs and CDFs and generate random variates for these two bivariate continuous distributions.

The PDF of Bivariate Normal Distribution. The PDF of the bivariate normal distribution is (17:492):

$$f(Y_1, Y_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho_{12}^2}} \exp\left\{ -\frac{1}{2(1-\rho_{12}^2)} \left[\left(\frac{Y_1-\mu_1}{\sigma_1}\right)^2 - 2\rho_{12}\left(\frac{Y_1-\mu_1}{\sigma_1}\right)\left(\frac{Y_2-\mu_2}{\sigma_2}\right) + \left(\frac{Y_2-\mu_2}{\sigma_2}\right)^2 \right] \right\} \quad (8)$$

TABLE 3

FUNCTIONS IN S AND MACROS IN THE ISTP FOR BIVARIATE CONTINUOUS DISTRIBUTIONS

Distributions	PDF	CDF	Random Variate
Bivariate Normal	?dbinorm	pnorm*pnorm (Independent)	?rbinorm
Bivariate Gamma (Independent only)	?gamfun*?gamfun	pgamma*pgamma	?rbigamma
		(Standard Gamma)	

where Y_1 and Y_2 are bivariate normal random variables; μ_1 and σ_1 are, respectively, the mean and the standard deviation of the marginal distribution of Y_1; μ_2 and σ_2 are, respectively, the mean and standard deviation of the marginal distribution of Y_2; ρ_{12} is the coefficient of correlation between random variables Y_1 and Y_2. Using Equation (8), one can accommodate the computations of the PDF of the bivariate normal distribution for both independent and dependent random variables. By replacing ρ_{12} with 0, Equation (8) can be used to compute the PDF of the bivariate normal distribution for independent random variables; and by giving any number between -1 and 1 except 0, Equation (8) can be used to compute the PDF of bivariate normal distributions for dependent random variables.

The CDF of Bivariate Normal Distribution. The computation of the bivariate normal CDF for two dependent random variables requires extensive amounts of time in the current S environment because it involves double integration using other packages such as Macsyma and IMSL. Evaluating double integration is very time consuming because it requires complex iterative computations. Hence, the CDF construction was limited to the independent case only. The CDF of the bivariate normal distribution for independent random variables is computed by multiplying the CDFs of two independent random variables.

Generating Random Variates for Bivariate Normal Distribution. Random variates from the bivariate normal distribution can be used for simulations or regression analysis. An easy way to generate random variates for the multivariate normal distribution is introduced as follows:

> Suppose we would like to generate a sample of 100 points from a multivariate normal distribution, with a specified correlation matrix, rho, a vector of standard-deviations for the variables, sd, and a vector of means, means. A good way to accomplish this starts with the correlation matrix, formed by multiplying the elements of the correlation matrix by the product of the row and column standard deviations. The sample itself is generated first as 100 samples from the standard multivariate normal. Then we multiply by the Choleski decomposition of the covariance matrix and add on the means.
>
> ```
> > var.cov_rho * outer(sd,sd)
> > x_matrix(rnorm(100*len(sd)),100,len(sd))
> > sample_x%*chol(var.cov)+
>          matrix(means, 100, len(sd), byrow=T).
> ```
>
> (5:211)

The PDF of Bivariate Gamma Distribution. Since the ISTP provides the macro ?gamfun for computing the PDF of the gamma random variable with two parameters, the bivariate PDF of independent gamma random variables can be computed by multiplying two independent PDFs.

The CDF of Bivariate Gamma Distribution. The bivariate CDF of two independent standard gamma random variables can be computed by means of the pgamma S function. As in the case for the bivariate CDF of normal independent random variables, the bivariate CDF of

independent gamma random variables is computed by multiplying their separate CDFs.

Generating Random Variate for Standard Bivariate Gamma Distribution. Independent random variates can be generated for the standard bivariate gamma distribution, because the CDF is computed for the standard bivariate gamma distribution for the case of independent random variables. A step-by-step procedure for generating independent random variates for the standard bivariate gamma distribution is now illustrated, where $F(x,y)$ is the CDF of the standard bivariate gamma distribution for independent random variables X and Y, and $F(x)$ and $F(y)$ are the CDFs of standard gamma distributions for random variables X and Y respectively:

Step 1. Generate random numbers, R, between 0 and 1.

Step 2. Use R as CDF of standard bivariate gamma distribution for independent random variables X and Y. This can be expressed as follows:

$$F(x,y) = R$$

Then the CDF of standard gamma distribution for random variables X, $F(x)$, can be arranged as follows:

$$R \leq F(x) \leq 1$$

Step 3. Generate uniform random variates Q, ranging between R and 1, then use these Q as the CDF of standard gamma distribution of random variables X.

$$R \leq Q \leq 1$$

$$F(x) = Q$$

Step 4. Since CDF of standard bivariate gamma distribution for independent random variables X and Y is the product of their separate CDFs, CDF of standard gamma distribution for random variables Y can be defined as follows:

$$F(x,y) = F(x) \quad F(y)$$

$$F(y) = F(x,y) \, / \, F(x)$$

$$= R \, / \, Q$$

Step 5. From the obtained cumulative probabilities of random variables X and Y, the random variables can be found as follows:

$$X = \mathrm{qgamma}(Q, \alpha_1)$$

$$Y = \mathrm{qgamma}(R/Q, \alpha_2)$$

where α_1 and α_2 are parameters for standard gamma distributions.

<u>Plotting the Shape of Probability Distributions</u>. Using the existing S functions and macros of the ISTP which compute PDFs, PMFs, or CDFs for probability distributions, the shapes of probability distributions can be plotted in two-dimensional space for the univariate case and in three-dimensional space for the bivariate case.

For univariate probability distributions, up to four distributions can be displayed simultaneously with different parameters so that direct comparison among family members can be made.

Shapes of bivariate continuous distributions are displayed by using the PDF and CDF, respectively. The CDF can only be obtained for the bivariate normal distributions of independent random variables and the standard bivariate

gamma distribution for independent random variables, while PDFs can be obtained for the bivariate normal distribution for both independent and dependent random variables and both standard (one-parameter) and two-parameter bivariate gamma distributions for independent random variables.

The shape of PDFs is plotted in three-dimensional space accompanied with a contour plot in two dimensions. Shapes of CDFs are limited to the display of bivariate normal distributions for independent random variables and standard bivariate gamma distributions for independent random variables. Since these distributions are displayed in three-dimensional space, comparison among the family members with different parameters could not be accommodated.

Although plotting the shape of probability distributions involves elementary graphical functions of S such as plot(x,y) for a two-dimensional plot, contour(x,y,z) for a contour plot, and persp(z) for a three-dimensional plot, it facilitates imaging probability distributions and enhances the understanding of probability distributions by allowing comparisons of members of various families of distributions. However, comparison across distributions is not currently available.

Table 4 lists available macros covered by the ISTP allowing the graphics display of probability distributions.

TABLE 4

MACROS IN THE ISTP FOR PLOTTING THE SHAPE OF PROBABILITY DISTRIBUTIONS

UNIVARIATE DISTRIBUTIONS		(?uvd)	Plot PDF / PMF	Comparison PDF /PMF
	Binomial	?binom	?binplot	?obinplot
Discrete	Neg-Binomial	?negbinom	?negbinplot	?onegbinplot
Distribution	Hypergeometric	?hypergeo	?hyperplot	?ohyperplot
(?dis)	Uniform	?uniform	?unifplot	?ounifplot
	Poisson	?poisson	?poisplot	?opoisplot
	Normal	?normal	?normplot	?onormplot
	Uniform	?cuniform	?cunifplot	?ocunifplot
	Log-Normal	?lognorm	?lnormplot	?olnormplot
Continuous	T	?tdis	?tplot	?otplot
Distribution	Exponential	?expon	?explot	?oexplot
(?cont)	Gamma	?gammaset	?gamplot	?ogamplot
	Weibull	?weibull	?weibplot	?oweibplot
	Beta	?betaset	?betaplot	?obetaplot
	Chisquare	?chisquare	?chisqplot	?ochisqplot
	F	?fdis	?fplot	?ofplot

BIVARIATE CONTINUOUS DISTRIBUTIONS	(?bvd)	Plot PDF	Plot CDF
Bivariate Normal	?binormal	?binorm	?bicnorm
Independent Bivariate Gamma	?bigammaset	?bigamma	?bicgamma

Maximum Likelihood Estimation

Traditionally, too much emphasis has been placed on the computational mechanics of obtaining maximum likelihood estimations. The ISTP focuses alternatively on the nature of the process of maximum likelihood estimation by allowing the user to participate in the process of optimization and to observe the physical consequence of parameter choices.

Graphics Method of Maximum Likelihood Estimation. ISTP's procedure for maximum likelihood estimation assumes a random sample of size one or two has been obtained from a univariate discrete or univariate continuous population. The likelihood function is graphically presented and the user is asked to make a guess at what value the parameter must assume for the likelihood function to reach its peak. The user can run the program as many times as he wishes. After he observes the consequences of his parameter choice on any given run, he can request the true value of the maximum likelihood estimate (MLE) be displayed. By graphically participating in the act of optimization he receives visual confirmation that the MLE is that value of the parameter that maximizes the likelihood of occurrence of the sample. Once the student perceives the need for a heuristic to direct the search process, the instructor can suggest to the student that he consider taking advantage of information contained in the sample. This might

mean encouraging the student to compute estimates based on functions of the sample values. If the student achieves optimization using a particular function, with different samples, he will be led to the realization that he has discovered for himself an estimator that is the MLE. Such a discovery can motivate a user's desire to look for a conceptual analogue of what is now an intuitively-derived notion.

Tables 5 and 6 list the available macros for maximum likelihood estimation.

Transformation of Random Variables

Invariably, presentations in mathematical statistics rely heavily upon the calculus as a vehicle for teaching fundamental concepts of probability. This is especially true when the notion of the transformation of a random variable is introduced. This causes the student to focus on the mechanics of the process while providing little or no motivation for an intuitive grasp of the subject matter. The transformation methodology employed by the ISTP, on the other hand, does not require mastery of such exhaustive mathematics. The student constructs the density of the transformed variable by using the properties of the density function of a random variable. This allows students to concentrate their attention on the iterative mapping of the regions under the original PDF into their corresponding regions under the PDF of the transformed variable. This

TABLE 5

MACROS IN THE ISTP FOR MAXIMUM LIKELIHOOD
ESTIMATION OF DISCRETE DISTRIBUTIONS

<table>
<tr><th rowspan="2">Discrete Distributions (?mledisc)</th><th rowspan="2">Type of Parameters</th><th colspan="2">Sample Size 1</th><th colspan="2">Sample Size 2</th></tr>
<tr><th>1 para</th><th>2 para</th><th>1 para</th><th>2 para</th></tr>
<tr><td>Bernoulli (?mleber)</td><td>P of success</td><td>?mleber1</td><td></td><td>?mleber2</td><td></td></tr>
<tr><td>Poisson (?mlepos)</td><td>λ</td><td>?mlepos1</td><td></td><td>?mlepos2</td><td></td></tr>
<tr><td>Geometric (?mlegeo)</td><td>P of Success</td><td>?mlegeo1</td><td></td><td>?mlegeo2</td><td></td></tr>
<tr><td rowspan="3">ReyLam (?mlerey)</td><td></td><td colspan="2">?mlerey1</td><td colspan="2">?mlerey2</td></tr>
<tr><td>P1: P of x=0</td><td>?mlerey1p1</td><td rowspan="2">?mlerey1pb</td><td>?mlerey2p1</td><td rowspan="2">?mlerey2pb</td></tr>
<tr><td>P2: P of x=1</td><td>?mlerey1p2</td><td>?mlerey2p2</td></tr>
<tr><td rowspan="2">Uniform (?mledunif)</td><td>Lower bound</td><td></td><td rowspan="2"></td><td>?mleduifd1</td><td rowspan="2">?mledunifb</td></tr>
<tr><td>Upper bound</td><td></td><td>?mledunifu</td></tr>
</table>

TABLE 6

MACROS IN ISTP FOR MAXIMUM LIKELIHOOD ESTIMATION OF CONTINUOUS DISTRIBUTIONS

Continuous Distributions (?mlecont)	Type of Parameters	Sample Size 1		Sample Size 2	
		1 para	2 para	1 para	2 para
Normal (?mlenorm)		?mlenorml		?mlenorm2	
	Mean	?mlenormlm	?mlenormlb	2mlenorm2m	?mlenorm2b
	Standard Dev	?mlenormls		2mlenorm2s	
Log-Normal (?mlelnorm)		?mlelnorml		?mlelnorm2	
	Mean	?mlelnormlm		?mlelnorm2m	?mlelnorm2b
	Standard Dev	?mlelnormls		?mlelnorm2s	
Exponential (?mleexpon)	λ	?mleexpl		?mleexp2	
Uniform (?mlecunif)	Lower bound				?mlecunif
	Upper bound				
Standard Gamma (?mlestgam)	α	?mlestgaml		?mlestgam2	

ensures the concept of the transformation of a random variable remains the focal point of the presentation.

Graphical PDF Method of Variable Transformation. Random variables can be transformed by means of their PDFs. A step-by-step procedure for the PDF method of transformation follows using Y = Log(X), where X is a normally distributed random variable whose mean is μ and whose standard deviation is σ.

Step 1. Plot the transformation Y = Log(X) to discover if it is one-to-one or not, as shown in Figure 2.

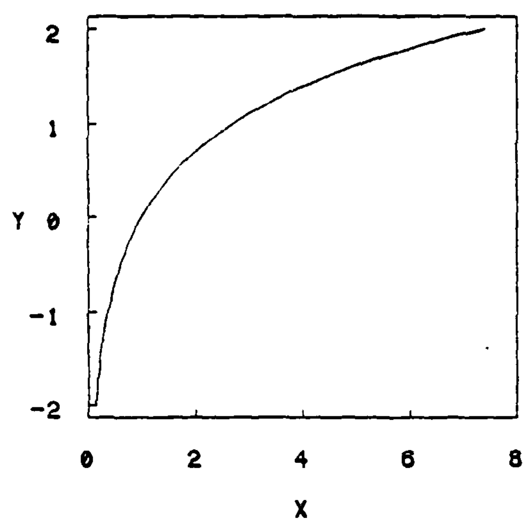


Fig. 2. Plot of Transformation Y = Log(X)

Step 2. Accept input to determine the range of Y, and Δy: the length of the sub-intervals into which the range is to be divided.

Accept : $Ys \leq Y \leq Ye$

Accept : Sub-interval Δy length

Let $Yi = Ys + \Delta y \cdot (i-1)$

Step 3. Compute the x_i and x_{i+1} corresponding to y_i and y_{i+1}.

$$x_i = \exp(y_i)$$

$$x_{i+1} = \exp(y_{i+1})$$

Step 4. Compute CDFs at x_{i+1} and x_i.

$$F(x_{i+1}) = \int_{-\infty}^{x_{i+1}} f(x)\, dx$$

$$F(x_i) = \int_{-\infty}^{x_i} f(x)\, dx$$

where $f(x) \sim N(\mu, \sigma)$

Step 5. Compute the area under $f(x)$ corresponding to the sub-interval (t_i, t_{i+1}) which is the area under $f(y)$ corresponding to the interval (y_i, y_{i+1}).

$$F(y_{i+1}) - F(y_i) = P(y_i \leq Y \leq y_{i+1})$$

$$= P(x_i) \leq X \leq x_{i+1}) = F(x_{i+1}) - F(x_i)$$

Step 6. Estimate the PDF of $Y{:}f(y)$ at y_i by dividing the area under $f(y)$ corresponding to interval (y_i, y_{i+1}) by Δy.

$$f(y_i) = \frac{F(y_{i+1}) - F(y_i)}{\Delta y}$$

Note that:

$$\lim_{\Delta y \to 0} \frac{F(y_{i+1}) - F(y_i)}{\Delta y} = F'(y_i)$$

$$\lim_{\Delta y \to 0} \frac{F(y_i + \Delta y) - F(y_i)}{\Delta y} = F'(y_i) = f(y_i)$$

This means as $\Delta y \to 0$, the estimate of the density of y_i approaches the true value of its PDF.

Analogous methods are applied to the univariate discrete random variables and bivariate random variables. Note that for the case of the univariate discrete random variables, the PDF of the original random variables can be transformed directly to the PDF of the corresponding transformed variables.

For the case of bivariate continuous random variables, the CDF of the original random variables cannot be obtained from direct S computation so that double integration routines in the IMSL library are used to estimate the CDF.

A PDF to PDF transformation without involving the estimation process called the method of distribution functions (16:236), is applied to the case of $Y = a \cdot X + b$ for the standard normal, the chi-square and the F random variables.

Table 7 lists the available macros for transformations of random variables of univariate distributions and bivariate distributions.

Hypothesis Testing

The cook-book orientation of standard presentations of hypothesis testing ignores the theoretical justification for selecting test statistics and addresses the concept of testing in a static fashion. Students are asked

TABLE 7

MACROS IN THE ISTP FOR TRANSFORMATION OF RANDOM VARIABLES

Transformation of Random Variables (?tran) — Type of Distributions for Random Variables			Macros	Available Transformation Form
Univariate (?uvtran)	Discrete (?trbin)	Binomial	?trbin	$Y = aX^2 + bX + c$
	Continuous (?trcont)	Standard Normal	?trstnorm	$Y = aX + b$
		Normal	?trnorm	$Y = X^2$ $Y = \text{Log}\ (X)$ $Y = \text{Exp}\ (X)$ $Y = \sqrt{X}$
		Uniform	?trunif	$Y = X^2$ $Y = \text{Log}\ (X)$ $Y = \text{Exp}\ (X)$ $Y = \sqrt{X}$
		Log-Normal	?trlnorm	$Y = X^2$ $Y = \text{Log}\ (X)$ $Y = \text{Exp}\ (X)$ $Y = \sqrt{X}$
		Exponential	?trexp	$Y = X^2$ $Y = \text{Log}\ (X)$ $Y = \text{Exp}\ (X)$ $Y = \sqrt{X}$

TABLE 7--Continued

Transformation of Random Variables (?tran)			Macros	Available Transformation Form
Type of Distributions for Random Variables				
Univariate (?uvtran)	Continuous (?trcont)	Standard Gamma	?trgamma	$Y = X^2$ $Y = \text{Log}\ (X)$ $Y = \text{Exp}\ (X)$ $Y = \sqrt{X}$
		F	?trf	$Y = X^2$ $Y = \text{Log}\ (X)$ $Y = \text{Exp}\ (X)$ $Y = \sqrt{X}$ $Y = aX + b$
		T	?trt	$Y = X^2$ $Y = \text{Log}\ (X)$ $Y = \text{Exp}\ (X)$ $Y = \sqrt{X}$
		Weibull	?trweib	$X = X^2$ $Y = \text{Log}\ (X)$ $Y = \text{Exp}\ (X)$ $Y = \sqrt{X}$
		Chisquare	?trchisq	$Y = X^2$ $Y = \text{Log}\ (X)$ $Y = \text{Exp}\ (X)$ $Y = \sqrt{X}$ $Y = aX + b$

TABLE 7--Continued

Transformation of Random Variables (?tran) — Type of Distributions for Random Variables		Macros	Available Transformation Form
Bivariate Continuous (?bvtran)	Normal (?bvntran)	?bvn1tran	$Z = X + Y$
		?bvn2tran	$Z = X^2 + Y^2$
	Uniform (?bvutran)	?bvu1tran	$Z = X + Y$
		?bvu2tran	$Z = X^2 + Y^2$
	Chisquare	?bvctran	$Z = (X/dfx) / (Y/dfy)$
	St-Normal & Chisquare	?bvnctran	$T = X / \sqrt{Yp/dfy}$

to accept procedures without justification and have little opportunity to explore the dynamic effect that sample size, type I error (α) and type II error (β), and alternative values of the parameters under investigation can have on the power of such tests. The ISTP's graphical approach to testing establishes the credibility for using one test statistic over others by offering the student a graphical confirmation that values of the test statistic are related on a one-to-one basis with values of the likelihood ratio. Furthermore, a dynamic and revisable plot of the power function is displayed simultaneously with a plot representing the sampling distribution of the test statistic. With access to both functions the student can learn to appreciate the rationale for selecting test statistics. Most importantly, he has an opportunity to observe dynamically the effects of varying each critical aspect of testing.

The prototypical version of the ISTP contains macros to facilitate the process for the test of a mean of a normal random variable when σ is known and when σ is unknown. Power computations using standard normal and non-central distributions are made and graphed for observation by the user.

Graphical Method of Hypothesis Testing. This prototypical version of the ISTP executes the following

steps to perform a hypothesis test for a mean of a normal distribution when σ is known:

1. Take input values proposed for the null and alternative hypotheses.

2. Compute critical values in terms of $\overline{X}$ and Z for rejecting H_o.

3. Compute the likelihood ratio statistic associated with the critical value of $\overline{X}$.

4. Compute the type II error and power function.

5. Compute the test statistic Z or use the given sample mean as the test statistic.

6. Compare the test statistic to its critical value to decide whether H_o should be rejected or not.

7. Illustrate steps 1 through 6 on four screens as follows:

   a. Plot the null hypothesis and alternate hypothesis on the $\overline{X}$ scale.

   b. Plot the likelihood ratio function and the sampling distribution of the null and alternative distributions on the $\overline{X}$ scale.

   c. Plot the power function and show the rejection area on the Z scale.

   d. Show the factors involved with decision making by comparing the test statistic $\overline{X}$ to its critical value or comparing the test statistic Z to its critical value and provide a prob value to indicate the significance of the test.

To perform the hypothesis test for a mean of normal distribution when σ is unknown, the following steps are employed:

1. Take input values proposed for null and alternative hypotheses.

2. Compute a critical value of T for rejecting H_o.

3. Compute the value of the likelihood ratio statistic associated with the critical value of T.

4. Compute the type II error and power function.

5. Compute the test statistic T.

6. Compare the test statistic to its critical value to decide whether H_o should be rejected or not.

7. Illustrate steps 1 through 6 on three screens as follows:

   a. Plot the null hypothesis and alternative hypothesis on the $\overline{X}$ scale.

   b. Plot the likelihood ratio function and show the rejection area on the T scale.

   c. Plot the power function and show the factors involved with decision making by comparing the test statistic to its critical value and provide a prob value to indicate the significance of the test.

Table 8 lists the available macros in the ISTP for conducting these two hypothesis tests.

<u>Likelihood Ratio Statistic</u>. The likelihood ratio statistic (LRS) can be computed by using Equation (9).

$$\text{LRS} = \frac{\text{maximum likelihood, computed assuming Ho true}}{\text{maximum likelihood, computed assuming Ha true}} \qquad (9)$$

If the likelihood ratio is sufficiently large, H_o should be accepted, because when H_o is true the value for the numerator will be large so as to make the value of LRS large. Therefore, the likelihood ratio statistic can be used as another test statistic compared to its critical value.

TABLE 8

MACROS IN THE ISTP FOR HYPOTHESIS TESTING

Test of Mean of Normal Distribution (?hypo)	Alternate Hypothesis	Direction of Test	Macro
When σ is known (?test1)	$\mu > \mu$	Right tail	?test1r
	$\mu < \mu$	Left tail	?test1l
	$\mu \neq \mu$	Two tail	?test1t
When σ is unknown (?test2)	$\mu > \mu$	Right tail	?test2r
	$\mu < \mu$	Left tail	?test2l
	$\mu \neq \mu$	Two tail	?test2t

NOTE: Test for the Mean of Normal Distribution.

When H_o is true, μ equals μ_o. Thus, μ would be replaced by μ_o in the joint PDF for the normal distribution. When H_a is true, μ would be replaced by $\overline{X}$ because $\overline{X}$ is the MLE for μ. These conditions yield the following results:

When σ is known, the likelihood ratio λ_n is:

$$\lambda_n = \frac{\dfrac{1}{(2\pi\sigma^2)^{n/2}} \cdot e^{\dfrac{-\Sigma(X_i-\mu_o)^2}{2\sigma^2}}}{\dfrac{1}{(2\pi\sigma^2)^{n/2}} \cdot e^{\dfrac{-\Sigma(X_i-\overline{X})^2}{2\sigma^2}}}$$

$$= e^{(2n\overline{X}\mu_o - n\mu_o^2 - n\overline{X}^2) / 2\sigma^2} \tag{10}$$

Equation (10) can be simplified as Equation (11):

$$\lambda_n = e^{-n(\overline{X}-\mu_o)^2 / 2\sigma^2} \tag{11}$$

And when σ is unknown, λ_n is:

$$\lambda_n = \frac{\dfrac{1}{\left(2\pi \cdot \dfrac{\Sigma(X_i-\mu_o)^2}{n}\right)^{n/2}} \cdot e^{-\frac{1}{2} \cdot \dfrac{\Sigma(X_i-\mu_o)^2}{\Sigma(X_i-\mu_o)^{2/n}}}}{\dfrac{1}{\left(2\pi \cdot \dfrac{\Sigma(X_i-\overline{X})}{n}\right)^{n/2}} \cdot e^{-\frac{1}{2} \cdot \dfrac{\Sigma(X_i-\overline{X})^2}{\Sigma(X_i-\overline{X})^{2/n}}}}$$

$$= \left[\frac{\Sigma(X_i-\bar{X})^2}{\Sigma(X_i-\mu_o)^2}\right]^{n/2}$$

$$= \left[1 + \frac{(\bar{X}-\mu_o)^2}{\frac{\Sigma(X_i-\bar{X})^2}{n}}\right]^{-n/2} \tag{12}$$

Equation (12) can be developed as Equation (13):

$$\lambda_n = \left[1 + \frac{(\bar{X}-\mu_o)^2}{\frac{\Sigma(X_i-\bar{X})^2}{n-1} \cdot \frac{1}{n}} \cdot \frac{1}{n-1}\right]^{-n/2}$$

$$= \left[1 + t^2 \frac{1}{n-1}\right]^{-n/2} \tag{13}$$

Therefore, Equation (11) can be used to obtain the likelihood ratio statistic of the test for the mean of a normal distribution with a known σ and Equation (13) can be used for the case of an unknown σ.

Actual examples of S macros being used to accomplish tests and procedures described in each of the four selected areas are presented in the next chapter.

IV. Using the ISTP and Output Interpretation

In general, there are two ways of accessing the ISTP: menu-driven selection and direct macro selection. Before discussing detailed procedures for using the ISTP, its general structure is introduced and preliminary descriptions of macros are provided.

Structure of the ISTP

The ISTP is an interactive statistical tutorial package consisting of four learning modules dealing with probability distributions, maximum likelihood estimation, transformation of random variables, and hypothesis testing. Figure 3 provides an overview of the structure of the ISTP. The ISTP has been constructed as a series of nested learning modules. This hierarchical structure allows the user to invoke the ISTP via menus or by direct macro selection.

Although Figure 3 provides a high-level presentation of the ISTP's anatomy, it does not reveal the names of lowest level macros. All such macros are listed in tables referred to by Figure 3. For example, from Figure 3, the user can determine that the module dealing with probability distributions contains macros covering univariate discrete distributions. Figure 3 indicates that if the user wants to know what univariate discrete distribution

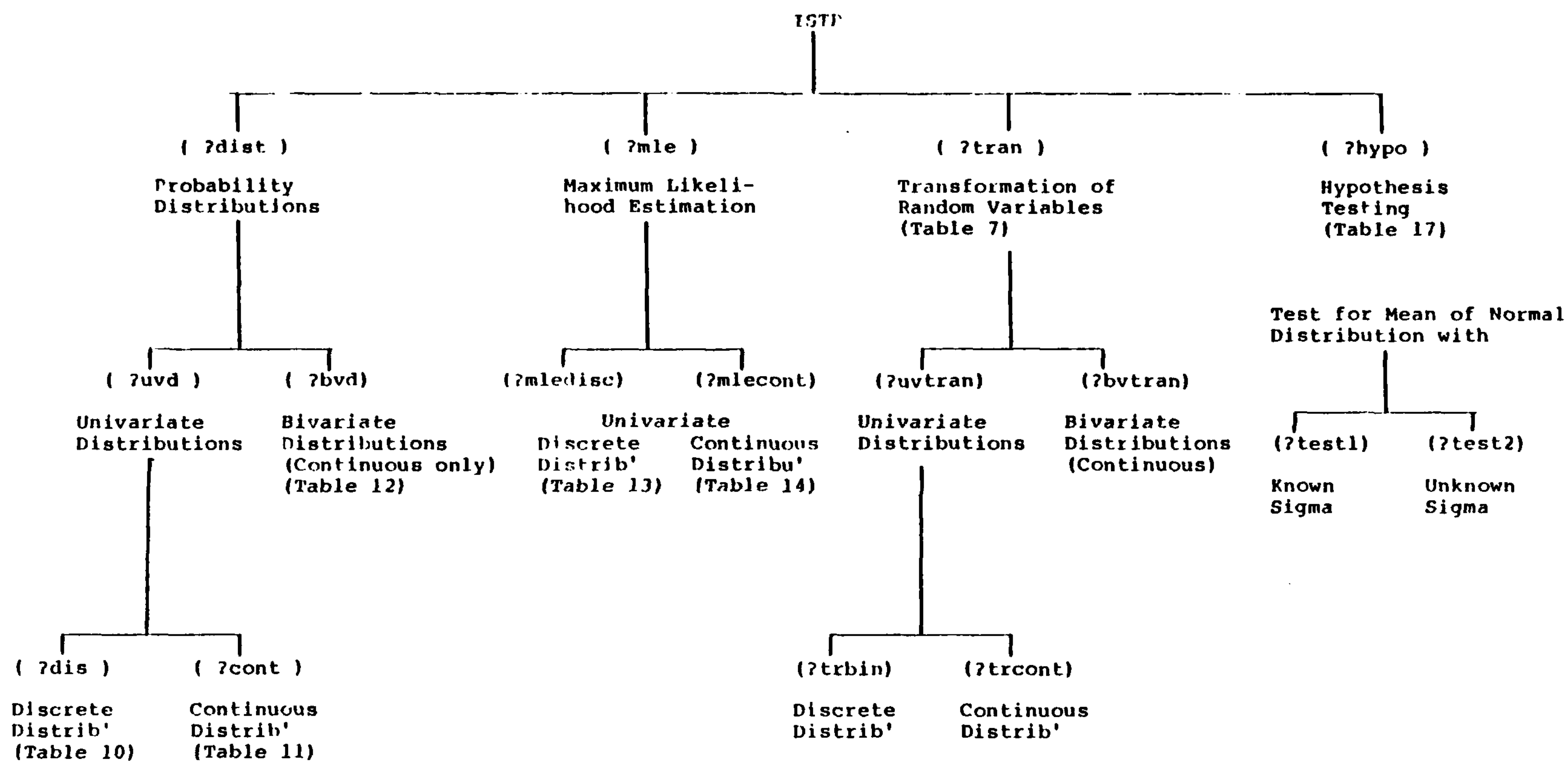


Fig. 3. Structure of the ISTP

macros the ISTP contains or what functions the ISTP provides for univariate discrete distributions, he should refer to Table 10.

Macros in S

S has three director levels: a user's working directory, a user's save directory, and the S system directory (5:47). Datasets created by S assignment statements and intermediate datasets are stored in the position 1 directory. The position 2 directory, the user's save directory, can be used to permanently store datasets. Macros defined by the user are automatically saved in this directory. The working directory and the save directory are owned by the current user. The S system directory, the position 3 directory, contains datasets and macros that can be accessed by all S users. An individual S user only has read and execute permission for items placed in the S system directory. Changes to the position 3 directory can only be made by the S super-user.

S macros are built from three parts: the MACRO statement, body, and END statement. The MACRO statement names the macro and its arguments; the body contains S expressions; and the END statement indicates the end of the macro. For example, a macro named mber could be constructed as in Figure 4.

The macro presented in Figure 4 was created to compute the PMF of a Bernoulli distribution. Once this

```
MACRO mber(
x/?PROMPT(X value; 0 or 1 : )/,
p/?PROMPT(P of X = 1      :)/)
#
#   This is a macro to compute the PMF for Bernoulli
    Distributions.
#
({
     ?T(x)_x
     ?T(p)_p
     ?T_(?T(p)^?T(x))*((1-?T(p))^(1-?T(x)))
     rm(?T(x), ?T(p), ?T, value=?T)
})
END
```

Fig. 4. Example of a Macro to Compute the PMF of a Bernoulli Distribution (Using Intermediate Datasets)

macro is defined in the user's save directory or S system directory, users can call this macro by issuing the command ?mber and supplying proper arguments, at any time. "Macros are introduced by the character "?", followed by the macro name and optional arguments in the parenthesis" (5:155). Without such macros, the user would have to compute the PMF of Bernoulli distributions by typing the PMF equation with actual values every time he wants it. The necessary sequence of S expressions to do this are listed in Figure 5.

```
> x_1
> p_.3
> pmf_(p^x)*((1-p)^(1-x))
> pmf
  0.3
```

Fig. 5. Example of Computing the PMF for a Bernoulli Distribution Without Using a Macro

The macro facility "allows new function-like operations to be built up from S expressions, and allows commonly used expressions to be saved and executed with a minimum of typing" (5:155).

Interaction between the user and the computer can be enhanced by using the macro ?PROMPT which executes if the user fails to provide the argument. This allows users to call the macro without giving arguments. In such an instance, the macro will ask the user to provide the proper arguments by printing the messages in parenthesis located next to the ?PROMPT.

Using intermediate datasets such as ?T(x) or ?T becomes important when the prompting option is executed by a macro. Unless the input variables are redefined by using such intermediate datasets, the macro ?PROMPT will cause S to ask the user to provide inputs every time the input variables are used in an S expression. For example, if the macro ?mber in Figure 6 is executed, S will ask the

```
MACRO mber(
x/?PROMPT(X value; 0 or 1 : )/,
p/?PROMPT(P of X = 1      : )/)
#
#  This is a macro to compute the PMF for Bernoulli
   Distributions.
#
({
    ?T_(p^x)*((1-p)^(1-x))
    rm(?T, value=?T)
 )}
END
```

Fig. 6. Example of a Macro to Compute the PMF of a Bernoulli Distribution (Without Using Intermediate Datasets)

user to provide inputs x and p two times, because variables x and p appear twice in the body of the macro.

Macros are compiled by using the S function define. Assuming the "pmfber" is the name of the UNIX file where the macro ?mber is stored, the user can execute the define command as follows:

```
> define ("pmfber")
  mac.mber
```

S responds to the define command by printing the macro name preceded by mac.. S stores the macro ?mber in file mac.mber in the user's save directory.

Methods of Accessing the ISTP

The ISTP is constructed hierarchically using a combination of menu functions and macros. As mentioned earlier, there are two general ways of accessing the ISTP: by menus and by direct macros selection. These options are documented by Figure 7. Figure 3 (page 61) provided the menu hierarchy. The tables referred to in Figure 3 contain the information needed to execute macros directly.

Most of the ISTP macros are designed to provide graphical output. Its remaining macros compute PDFs (PMFs), CDFs and quantiles or generate the random variates for probability distributions. Users should request activation of an appropriate driver (printer, tek10, or

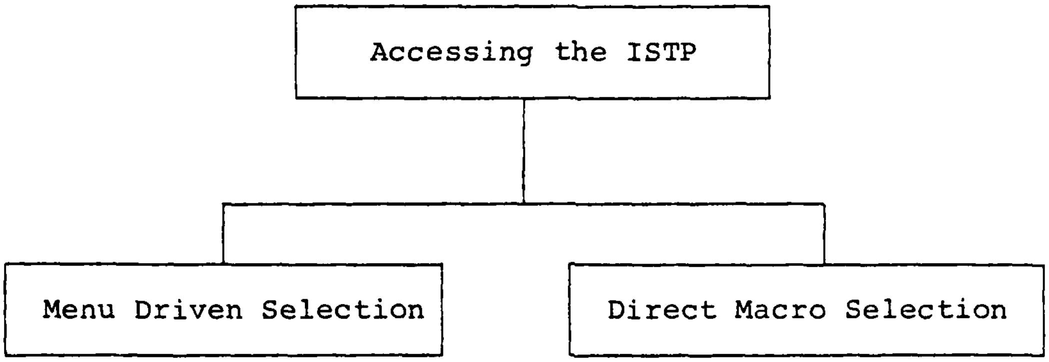


Fig. 7. Two Ways of Accessing the ISTP

hp7222v, etc.) before executing the ISTP macros when graphical results are expected.

Menu Driven Selection. Execution of any macro in the ISTP can be requested by calling one of the four highest macros: ?dist, ?mle, ?tran, or ?hypo or any macro superior to the desired macro in the hierarchy of macros, except the macros which compute PDFs (PMFs), CDFs or quantiles and generate random variates of univariate probability distributions.

It is recommended that beginners use the ISTP by calling one of the highest macros. The ISTP will then provide menus which allow users to select macros at lower levels. By using three or four second level menus in succession, the user can reach the macro that he wants to execute. For example to generate 20 random variates from the bivariate normal distribution, whose $\mu_x = 2$, $\sigma_x = 3$, $\mu_y = 2$, $\sigma_y = 1$, and whose correlation coefficient $\rho = 0.5$,

the user can invoke macro ?dist and request the following sequence of menus as displayed in Figure 9 on the next page.

When the user does not know the highest macro name, he can obtain the name by typing ?istp as in Figure 8.

```
  ?istp
1 - Probability Distribution
2 - Maximum Likelihood Estimation
3 - Transformation of Random Variables
4 - Hypothesis Test
Which one? 1

Please call macro '?dist' for
       the area of Probability Distributions
```

Fig. 8. Example of Getting Assistance for the ISTP

Direct Macro Selection. Menus require less training to use than the command language; however, "one of the major drawbacks to menu selection is in inefficiency for the expert user who wants to go directly to a specific command" (8:545-546). Therefore, menu-driven selection is recommended for beginners and direct macro selection is recommended for expert users.

Tables referred to in Figure 3 provide the information on direct macro selection except for the transformation random variables. The descriptions of the lowest level macros are provided in Table 10 through Table 14 and Table 17. (These tables are shown later in this chapter.) For example, Table 10 can be used to find out which distributions and macros are available for each univariate discrete distribution in the probability distribution area.

```
> ?dist

1 - Univariate distribution
2 - Bivariate distribution
Which one? 2

1 - NORMAL BIVARIATE DISTRIBUTION
2 - INDEPENDENT GAMMA BIVARIATE DISTRIBUTION
3 - QUIT
Which one? 1

1 - SHAPE OF BIVARIATE PDF
2 - SHAPE OF INDEPENDENT BIVARIATE CDF
3 - PDF
4 - INDEPENDENT CDF
5 - GENERATE RANDOM DEVIATES
6 - RETURN TO PREVIOUS STEP
Which one? 5

Number of sample  : 20
Mean of X         : 2
Sigma of X        : 3
Mean of Y         : 2
Sigma of Y        : 1
Correlation       : .5

Array:
2 by 20

           [, 1]        [, 2]        [, 3]       [, 4)      [, 5]

[1,]    0.2492454   5.075839   -0.849348   4.812448   1.267345
[2,]    1.590911    2.821804    1.111203   1.419085   2.121823

           [, 6]        [, 7]        [, 8]       [, 9]      [,10]

[1,]    5.417188   -7.066454    1.686742  -1.101984   3.634643
[2,]    1.891857   -0.6492715   1.092011   1.296121   2.324782

           [,11]        [,12]        [,13]       [,14]      [,15]

[1,]    1.874116    3.723961   -0.2604833  4.544921   3.106494
[2,]    3.968084    1.372114    1.560505   3.431603   2.224390

           [,16]        [,17]        [,18]       [,19]      [,20]

[1,]    4.754334    1.654156    0.0365566  1.792415   1.845809
[2,]    1.573221    2.927262    3.167427   0.4695425  1.206434
```

Fig. 9. Example of Invoking Menus

Table 10 indicates that the Bernoulli, the binomial, the negative binomial, the geometric, the hypergeometric, the uniform, the Poisson, and the ReyLam distribution are covered by the ISTP. Furthermore, Table 10 provides all the information related to direct macro selection. On the other hand, Table 7 does not provide the lowest level macros which the user could invoke for direct macro selection. In other words, if the user wants to graph the transformation of Y = Log(X) for normal random variables, he must call at least macro ?trnorm. This will generate menus he needs to gain access.

Available methods for accessing macros of the ISTP are summarized in Table 9.

As mentioned earlier, the macros which compute the PDF (PMF), the CDF and quantiles or generate random variates cannot be invoked by using menus except those associated with bivariate continuous distributions. Since all S functions compute PDFs, CDFs and quantiles or generate random variates without menu operations, the macros of the ISTP which perform similar functions for univariate distributions have been designed to be invoked in the same way.

As indicated by Figure 10, direct macro selection offers two access options: calling the macro with proper arguments as inputs or calling the macro without arguments. Since all macros require inputs from the user, the user should provide correct inputs by one of the following two

TABLE 9

AVAILABLE METHODS FOR ACCESSING MACROS OF THE ISTP

Topic Area			Method of Invoking	
			Menu Driven	Direct Select
Probability Distributions	Univariate	PDF (PMF), CDF Quantiles Generating R.V.	X	0
		Plot the Shape	0	0
	Bivariate	Plot the Shape Generating R.V./PDF*	0	0
Maximum Likelihood Estimation			0	0
Transformation of Random Variables			0	X
Hypothesis Testing			0	0

NOTES:

0: Available

X: Not Available

*: Only for bivariate normal.

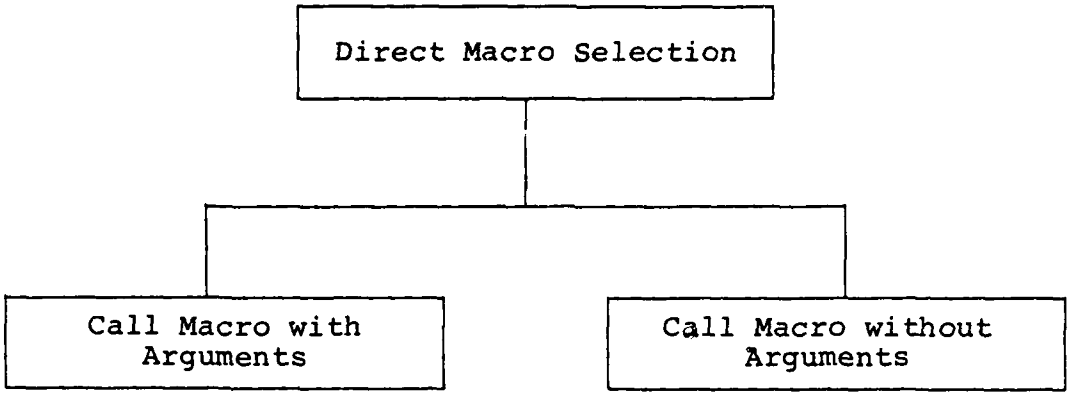


Fig. 10. Two Ways of Asking for Direct Macro Selection

ways. If the user knows the input requirements exactly, the user can call the macro providing arguments in the parenthesis following the macro name. On the other hand, if the user does not know the input requirement, the user can call the macro by issuing the macro name only. The macro will then use the prompting function to request required inputs. For example, to generate 10 random variates from the bivariate normal distribution with $\mu_x = 3$, $\sigma_x = 3$, $\mu_y = 2$, $\sigma_y = 2$ and whose correlation coefficient $\rho = 0.2$, either of two methods presented in Figure 11 can be exercised. Note: the same random number seed is given in both to guarantee identical results are achieved in each case.

The input requirements for most macros are as easy to specify as for the functions of S. On the other hand, there are some macros that are rather complicated to use.

```
> seed_Random.seed
> Random.seed_seed
#
# A certain number was given to the random number seed.
#

> ?rbinorm(10,3,3,2,2,.2)  # Direct macro selection with
                           # proper arguments as inputs.

Array:
2 by 10

          [, 1]     [, 2]     [, 3]      [, 4]     [, 5]
[1,]   1.288061  3.293183  1.975809  -2.119554  4.930751
[2,]   0.490316  3.465924  2.668956  -0.818479  5.483446

          [, 6]     [, 7]     [, 8]      [, 9]     [,10]
[1,[   3.478800  0.880869  1.926381   6.714127  1.254592
[2,]   3.147666  3.063315  2.591642   3.611894  2.210813

>Random.seed_seed
#
# Same random number seed was assigned.
#

>?rbinorm    # direct macro selection without arguments.

Number of sample   : 10
Mean of X          : 3
Sigma of X         : 3
Mean of Y          : 2
Sigma of Y         : 2
Correlation        : .2


Array:
2 by 10

          [, 1]     [, 2]     [, 3]      [, 4]     [, 5]
[1,]   1.288061  3.293183  1.975809  -2.119554  4.930751
[2,]   0.490316  3.465924  2.668956  -0.818479  5.483446

          [, 6]     [, 7]     [, 8]      [, 9]     (,10]
[1,]   3.478800  0.880869  1.926381   6.714127  1.254592
[2,]   3.147666  3.063315  2.591642   3.611894  2.210813
```

Fig. 11. Example of Direct Macro Selection Using Random Variates Generator

If the user is not familiar with the input requirement such as the input requirements for the macro dealing with plotting the shape of a distribution, the macro should be called without arguments. The ISTP will then tell the user what inputs to enter.

The rest of this chapter discusses the input requirements and how to interpret the output of macros contained in probability distribution, maximum likelihood estimation, transformation of random variables, and hypothesis testing modules.

Probability Distributions

Probability distributions in the ISTP can be used for two purposes: to compute PDFs (PMFs), CDFs or quantiles and generate random variates and to plot the shape of a particular distribution or to compare the shapes of distributions among a family's members. Macros which plot the shape of distributions can be called either via menu selection or by direct macro selection, while the macros which compute PDFs (PMFs), CDFs, or quantiles; or that generate random variates, can only be called by direct macro selections, except for the bivariate continuous distributions.

Input Requirements. Tables 10 through 12 provide detailed descriptions of the input requirements for all macros of probability distributions in the ISTP. The input

TABLE 10

DESCRIPTIONS OF MACROS IN THE ISTP FOR UNIVARIATE DISCRETE DISTRIBUTIONS

Discrete Distributions	Macros	Descriptions	Inputs
Bernoulli	?mber(x,p)	PMF	x : 0(failure) or 1(success) p : P of success m : number of R.V.
	?rber(m,p)	Generate R.V.	
Binomial	?binplot(n,p)	Plot shape	n(v) : number of trials p(v) : P of success m1 : number of plot(max 4) m2 : number of R.V. x : number of success x1 : min number of success x2 : max number of success
	?obinplot(m1,nv,pv)	Compare shape	
	?mbin(x,n,p)	PMF	
	?bin(x1,x2,n,p)	CDF	
	?rbin(m2,n,p)	Generate R.V.	
Negative-Binom'al	?negbinplot(r,p,mx)	Plot shape	r(v) : number of success p(v) : P of success m1 : number of plot(max 4) m2 : number of R.V. x : number of failure x1 : min number of failure x2 : max number of failure mx : max range for plot
	?onegbinplot (m1,rv,pv,mx)	Compare shape	
	?mnegbin(x,r,p)	PMF	
	?negbin(x1,x2,r,p)	CDF	
	?rnegbin(m2,r,p)	Generate R.V.	

TABLE 10--Continued

Discrete Distributions	Macros	Descriptions	Inputs
Geometric	?mgeo(x,p)	PMF	x : number of failures p : P of success x1 : min number of failure x2 : max number of failure m : number of R.V.
	?geo(x1,x2,p)	CDF	
	?rgeo(m,p)	Generate R.V.	
Hyper-Geometric	?hyperplot(n,M,N)	Plot shape	n(v) : sample size M(v) : success in population N(v) : population size m1 : number of plot(max 4) m2 : number of R.V. x : number of success x1 : min number of success x2 : max number of success
	?ohyperplot (m1,nv,Mv,Nv)	Compare shape	
	?mhyper(x,n,M,N)	PMF	
	?hyper(x1,x2,n,M,N)	CDF	
	?rhyper(m2,n,M,N)	Generate R.V.	
Uniform	?unifplot(l,u,mn,mx)	Plot shape	l(v) : lower bound u(v) : upper bound m1 : number of plot(max 4) m2 : number of R.V. x : quantiles mn : min range for plot mx : max range for plot
	?ounifplot(m1,lv,uv)	Compare shape	
	?ddunif(x,l,u)	PMF	
	?pdunif(x,l,u)	CDF	
	?rdunif(m2,l,u)	Generate R.V.	

TABLE 10--Continued

Discrete Distributions	Macros	Descriptions	Inputs
Poisson	?poisplot(l,mx)	Plot shape	l(v) : λ ml : number of plot(max 4) m2 : number of R.V. x : quantile xl : min quantile x2 : max quantile mx : max range for plot
	?opoisplot(ml,lv,mx)	Compare shape	
	?mpoiss(x,l)	PMF	
	?poiss(xl,x2,l	CDF	
	?rpoiss(m2,l)	Generate R.V.	
ReyLam	?mreylam(x,pl,p2)	PMF	x : 0, 1 or 2 pl : P of x = 0 p2 : P of x = 1 m : number of R.V.
	?rreylam(m,pl,p2)	Generate R.V.	

NOTES:

P: Probability

(v): Vector

TABLE 11

DESCRIPTIONS OF MACROS IN THE ISTP FOR UNIVARIATE CONTINUOUS DISTRIBUTIONS

Continuous Distributions	Macros	Descriptions	Inputs
Normal	?normplot (m,s,mn,mx)	Plot shape	m(v) : mean s(v) : standard deviation n : number of plot(max 4) mn : min range of plot mx : max range of plot
	?onormplot (n,mv,sv,mn,mx	Compare shape	
Uniform	?cunifplot (l,u,mn,mx)	Plot shape	l(v) : lower bound u(v) : upper bound n : number of plot(max 4) mn : min range of plot mx : max range of plot
	?ocunifplot (n,lv,uv)	Compare shape	
Log-Normal	?lnormplot (m,s,mx)	Plot shape	m(v) : mean s(v) : standard deviation n : number of plot(max 4) mx : max range of plot
	?olnormplot (n,mv,sv,mx)	Compare shape	
T	?tplot (df,mn,mx)	Plot shape	df(v): degree of freedom n : number of plot(max 4) mn : min range of plot mx : max range of plot
	?otplot (n,dfv,mn,mx)	Compare shape	

TABLE 11--Continued

Continuous Distributions	Macros	Descriptions	Inputs
Exponential	?explot(l,mx)	Plot shape	l(v) : λ nl : number of plot(max 4) n2 : number of R.V. mx : max range of plot x : quantile p : probability
	?oexplot(nl,lv,mx)	Compare shape	
	?dex(x,l)	PDF	
	?pex(x,l)	CDF	
	?qex(p,l)	Quantile	
	?rex(n2,l)	Generate R.V.	
Gamma	?gamplot (a,b,mx)	Plot shape	a(v) : α b(v) : β n : number of plot(max 4) mx : max range of plot x : quantile
	?ogamplot (n,av,bv,mx)	Compare shape	
	?gamfun(x,a,b)	PDF for 2 parameters	

TABLE 11--Continued

Continuous Distributions	Macros	Descriptions	Inputs
Weibull	?weibplot(a,b,mx)	Plot shape	a(v) : α (shape para) b(v) : β (scale para) n1 : number of plot(max 4) n2 : number of R.V. mx : max range of plot x : quantile p : probability
	?oweibplot (n1,av,bv,mx)	Compare shape	
	?dweib(x,a,b)	PDF	
	?pweib(x,a,b)	CDF	
	?qweib(p,a,b)	Quantile	
	?rweib(n2,a,b)	Generate R.V.	
Beta	?betaplot(a,b)	Plot shape	a(v) : α b(v) : β n : number of plot(max 4) mx : max range of plot(< 1)
	?obetaplot (n,av,bv,mx)	Compare shape	
Chisquare	?chisqplot(df,mx)	Plot shape	df(v): degree of freedom n : number of plot(max 4) mx : max range of plot
	?ochisqplot (n,dfv,mx)	Compare shape	
F	?fplot (df1,df2,mx)	Plot shape	df1(v): df of numerator df2(v): df of denominator n : number of plot(max 4) mx : max range of plot
	?ofplot (n,df1v,df2v,mx)	Compare shape	

NOTE: (v): Vector

TABLE 12

DESCRIPTIONS OF MACROS IN THE ISTP FOR BIVARIATE CONTINUOUS DISTRIBUTIONS

Bivariate Continuous	Macros	Descriptions	Inputs
Nommal Distribution	?binorm (mx,sx,my,sy,r,i)	Plot PDF	mx : μx sx : σx my : μy sy : σy r : correlation ρ i : intervals for plot x : quantile of x y : quantile of y n : number of R.V.
	?bicnorm (mx,sx,my,sy,i)	Plot CDF (Independent)	
	?dbinorm (mx,sx,my,sy,r,x,y)	PDF	
	?rbinorm (n,mx,sx,my,sy,r)	Generate R.V.	
Gamma Distribution (Independent)	?bigamma (ax,bx,ay,by,mx,i)	Plot PDF	ax : α for x bx : β for x ay : α for y by : β for y mx : max range for plot i : intervals for plot n : number of R.V.
	?bicgamma (ax,ay,mx,i)	Plot CDF (Standard)	
	?rbigamma(n,ax,ay)	Generate R.V.	

requirements are found in the parentheses next to the corresponding macro names. The inputs are explained in the input column. For example, the inputs for the macro ?mbin, which computes the PMF of the binomial distribution, are x, n, and p, where x is the number of successes, n is the number of Bernoulli trials, and p is the probability of success on a given trial. This is crucial information when the user wants to use the macro, with arguments, directly.

Another important input requirement needs to be considered if a comparison is to be made among a family's members. An example of calling a macro to compare four members of the normal family is displayed as Figure 12.

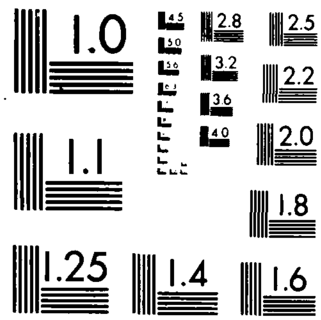
```
> ?onormplot

NUMBER OF PLOT(up to 4) : 4
MEAN (ex) 2. 3. 4.      : 0,0,-2,2
Std Dev (ex)1. 3. 7     : 1,2,3,4
MIN X for PLOT          : -6
MAX X for PLOT          : 6
```

Fig. 12. Example of Input Entry

As shown in Figure 12, the number of provided means and standard deviations is identical to the number of plots requested. In other words, if the user wants to compare four different normal distributions, he should specify 4 as the input for number of plots, and then provide four values for means and four values for standard deviations. The first mean should be paired with the first standard deviation and the second mean should be paired with the

AD-A174 297 DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL PACKAGE (ISTP) FOR LE..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF SYST.. K H CHUL 2/5

UNCLASSIFIED SEP 86 AFIT/GSM/ENC/86S-10 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

second standard deviation, and so on, so that the first plot will reflect a normal distribution with $\mu_1 = 0$ and $\sigma_1 = 1$; and the second one will reflect a normal distribution with $\mu_2 = 0$ and $\sigma_2 = 2$, etc. Even if the values of the means or standard deviations are the same, the user must specify every value explicitly. For example, if the user wants tc compare the shapes of three normal distributions whose means are all 1 and whose standard deviations are 1, 2, and 3 respectively, then the user should provide the input as in Figure 13.

```
> ?onormplot

NUMBER OF PLOT(up to 4) : 3
MEAN (ex) 2, 3, 4       : 1,1,1
Std Dev (ex)1, 3, 7     : 1,2,3
MIN X for PLOT          : -6
MAX X for PLOT          : 6
```

Fig. 13. Example of Input Entry

Caution should be exercised when the macro is invoked directly with arguments for comparing the shapes of distributions. Each input value should be entered as a vector of values using the c function of S as shown in Figure 14. However, the following type of entry must be avoided.

```
> ?onormplot(3,1,1,1,1,2,3,-6,6)
```

The input entry "MIN X for PLOT" and "MAX X for PLOT" provide the range for the plot. This needs to be entered by the user. By setting and resetting the range

```
> ?onormplot(3,c(1,1,1),c(1,2,3),-6,6)

 or

> mean c(1,1,1)
> stdv c(1,2,3)
> ?onormplot(3,mean,stdv,-6,6)
```

Fig. 14. Example of Input Entry

for a plot, the user can obtain different information from several plots of the same data set. For instance, by narrowing or expanding the range, the user can produce graphics that reflect zooming or panning over the data.

Output Interpretation. Figure 15 displays four univariate normal distributions, whose parameters are the same as those used in Figure 12. Each distribution has a corresponding number from 1 to 4 and each of their parameters is displayed in brackets at the bottom of the plot. Normal distribution one's mean and standard deviation are 0 and 1; normal distribution two's mean and standard deviation are 0 and 2; and so on.

Figure 16 displays four univariate binomial distributions whose parameters are placed above their plots. The scales of the x and y axes are arranged so that they can be compared easily.

Figures 17 and 18 display a Weibull distribution whose shape parameter is 7 and whose scale parameter is 1. The difference between these two figures can be accounted

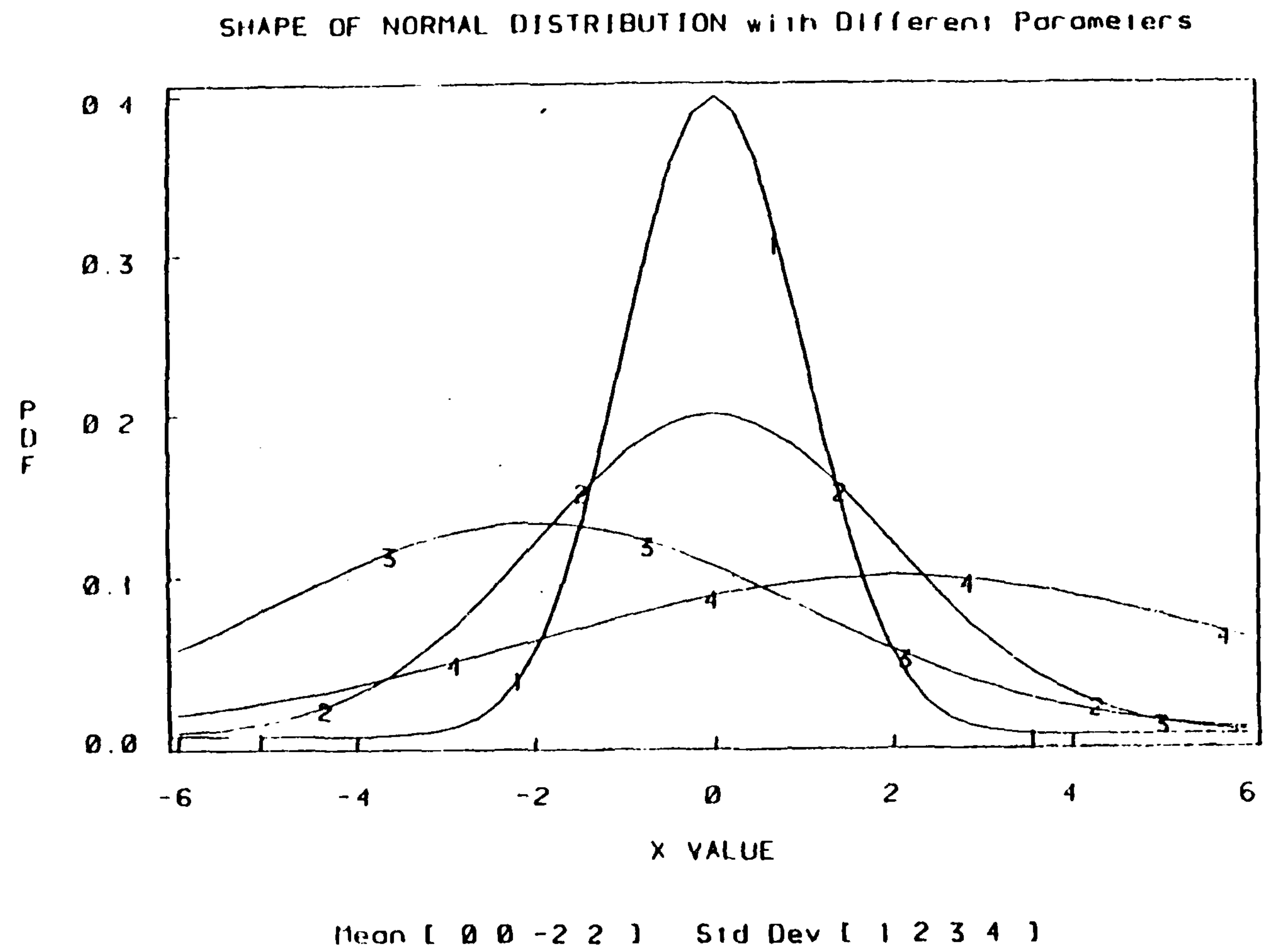


Fig. 15. Shapes of Normal Distributions with Different Parameters

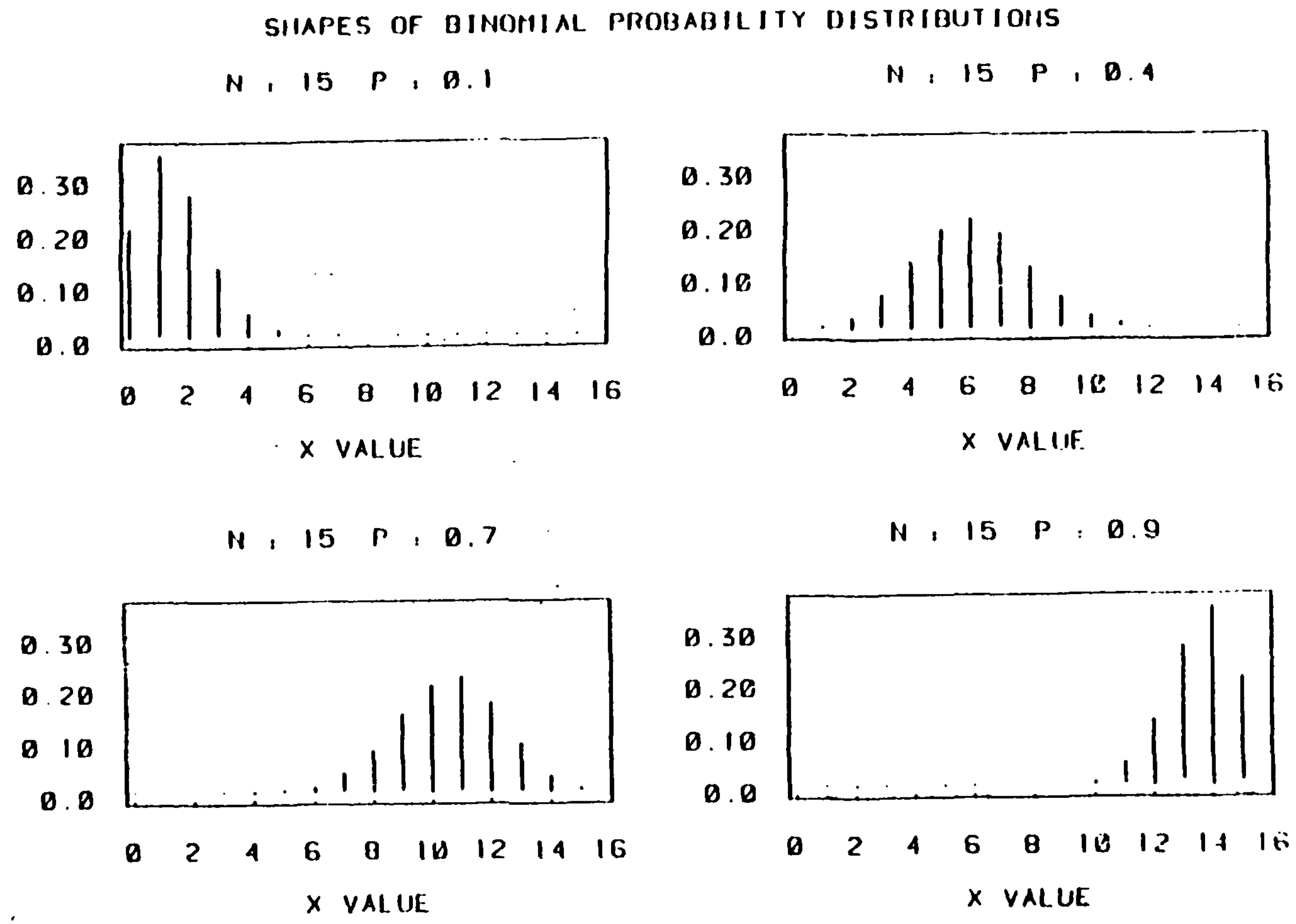


Fig. 16. Shapes of Binomial Distributions with Different Parameters

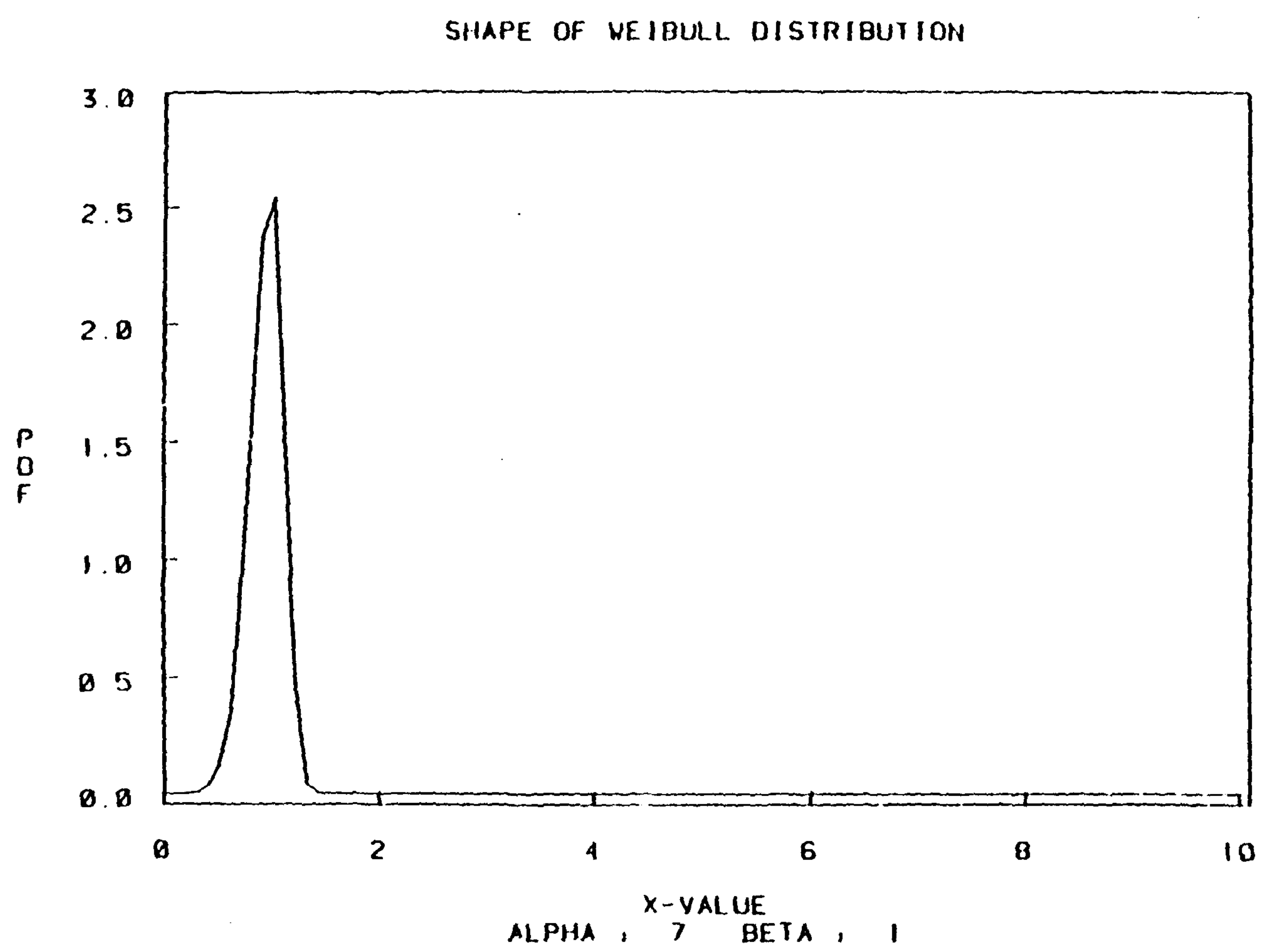


Fig. 17. Shape of the Weibull Distribution

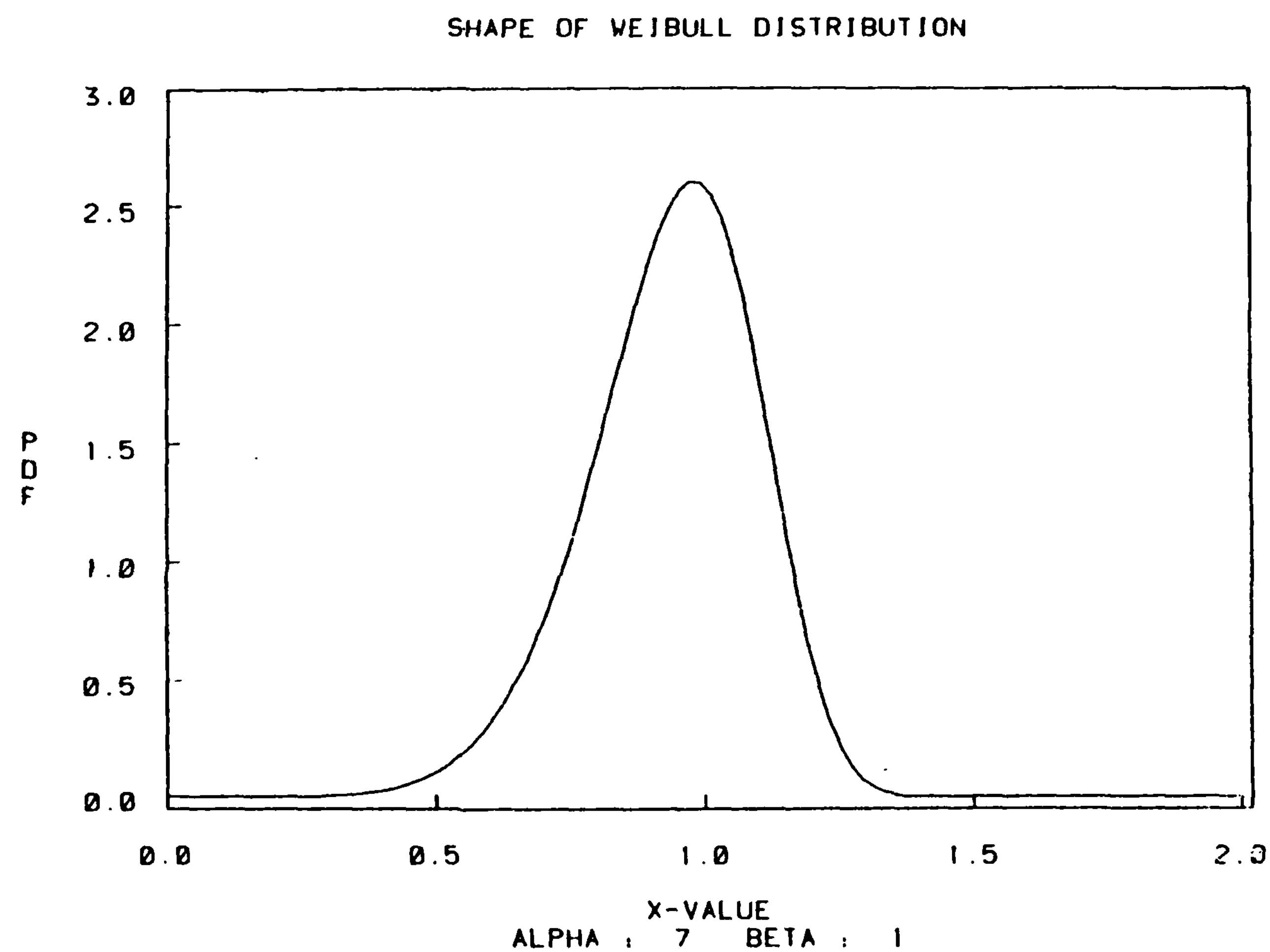


Fig. 18. Shape of the Weibull Distribution

for by their different ranges. A maximum plotting range of 10 is used for Figure 17, while 2 is used for Figure 18. Unlike the normal distribution, shapes of the Weibull or the gamma distribution for various parameters are not well known. Thus, it is usually difficult to select the proper range for a plot of these distributions. In this case, at first a wide range should be selected, as is done in Figure 17. By doing so the user can determine a reasonable range for the next plot. This will yield the favorable graphical result displayed in Figure 18.

Maximum Likelihood Estimation

When the maximum likelihood estimation macros are invoked, the user must provide actual samples of size one or two. Then the user extimates the parameter(s). The ISTP will provide two screens. On the first screen, the shape of the PDF (PMF) using the parameter(s) estimated by the user is plotted and the value of the PDF (PMF) for the given sample is provided. The second screen allows the user to observe the process of optimization by plotting the likelihood function against a suitable range of the parameter(s). The likelihood function is obtained by computing the joint PDF (PMF) of the given sample for these values of the parameter(s). Although the term jo t PDF (PMF) is not adequate for the case of sample size one, it is very useful when trying to graphically motivate the idea

of maximum likelihood estimation. If there are two parameters, as in the case of the normal distribution, both parameters can be estimated, or only one parameter can be estimated, if the other parameter is known and provided by the user.

The maximum likelihood estimation macros can be accessed by menu or by direct selection. When direct macro selection is desired, Tables 13 and 14 should be used to identify the input requirements.

Input Requirements. Table 15 provides information on what types of inputs are required when different sample sizes and numbers of parameters are selected. For example, if the sample size is two and the number of parameters is two with one known parameter, two sample values and one known parameter should be provided by the user. The user will also have to estimate the other parameter.

More detailed input requirements for maximum likelihood estimation are provided in Table 13 for univariate discrete distributions and Table 14 for continuous distributions. Table 14 indicates that the macro ?mlenorm2s requires five inputs: random variable x and y, correlation r, known mean m, and estimated standard deviation s. Although the macro will automatically compute the maximum likelihood estimate for the given sample x and y whose correlation is r and mean is m, the user is first asked

TABLE 13

DESCRIPTIONS OF MAXIMUM LIKELIHOOD ESTIMATION MACROS FOR UNIVARIATE DISCRETE DISTRIBUTIONS

Type	Macros	Descriptions	Inputs
Bernoulli	?mleber1 (x,ep)	1 sample	x : random variable 0 or 1 y : random variable 0 or 1 (e)p : P of r.v. = 1
	?mleber2 (x,y,ep)	2 sample	
Poisson	?mlepos1 (x,el)	1 sample	x : random variable ($\leq$30) y : random variable ($\leq$30) (e)l : λ
	?mlepos2 (x,y,el)	2 sample	
Geometric	?mlegeo1 (x,ep)	1 sample	x : random variable ($\leq$30) y : random variable ($\leq$30) (e)p : P of success
	?mlegeo2 (x,y,ep)	2 sample	

TABLE 13--Continued

Type	Macros	Descriptions	Inputs
ReyLam	?mlerey1p1 (x,p2,ep1)	1 sample for p1	x : random variable 0,1,or 2 y : random variable 0,1,or 2 (e)p1 : P of r.v. = 0 (e)p2 : P of r.v. = 1
	?mlerey1p2 (x,p1,ep2)	1 sample for p2	
	?mlerey1pb (x,ep1,ep2)	1 sample for p1 and p2	
	?mlerey2p1 (x,y,p2,ep1)	2 sample for p1	
	?mlerey2p2 (x,y,p1,ep2)	2 sample for p2	
	?mlerey2pb (x,y,ep1,ep2)	2 sample for p1 and p2	
Uniform	?mledunifl (x,y,u,el)	2 sample for l	x : random variable y : random variable (e)l : lower bound (e)u : upper bound
	?mledunifu (x,y,l,eu)	2 sample for u	
	?mledunifb (x,y,el,eu)	2 sample for l and u	

NOTES: P: Probability; (e): estimated by the user.

TABLE 14

DESCRIPTIONS OF MAXIMUM LIKELIHOOD ESTIMATION MACROS FOR UNIVARIATE CONTINUOUS DISTRIBUTIONS

Types	Macros	Descriptions	Inputs
Normal	?mlenorm1m(x,s,em)	1 sample for m	x : random variable y : random variable r : correlation (e)m : mean (e)s : standard deviation
	?mlenorm1s(x,m,es)	1 sample for s	
	?mlenorm1b(x)	1 sample for m&s	
	?mlenorm2m(x,y,r,s,em)	2 sample for m	
	?mlenorm2s(x,y,r,m,es)	2 sample for s	
	?mlenorm2b(x,y,r,em,es)	2 sample for m&s	
Log-Normal	?mlelnorm1m(x,s,em)	1 sample for m	x : random variable y : random variable (e)m : mean (e)s : standard deviation
	?mlelnorm1s(x,m,es)	1 sample for s	
	?mlelnorm2m(x,y,s,em)	2 sample for m	
	?mlelnorm2s(x,y,m,es)	2 sample for s	
	?mlelnorm2b(x,y,em,es)	2 sample for m&s	
Exponential	?mleexp1(x,el)	1 sample for l	x : random variable y : random variable (e)l : λ
	?mleexp2(x,y,el)	2 sample for l	

TABLE 14--Continued

Types	Macros	Descriptions	Inputs
Uniform	?mlecunif(x,y,el,eu)	2 sample for l&u	x : random variable y : random variable (e)l : lower bound (e)u : upper bound
Stand-Gamma	?mlestgam1(x,ea)	1 sample for a	x : random variable y : random variable (e)a: α
	?mlestgam2(x,y,ea)	2 sample for a	

NOTE: (e): estimated by the user.

TABLE 15

INPUT REQUIREMENTS FOR MAXIMUM LIKELIHOOD ESTIMATION MACROS

Sample Size	No. of Parameters	No. of Known Parameters	Input
1	1	0	1 sample 1 (e) parameter
	2	0	1 sample 2 (e) parameter
		1	1 sample 1 (e) parameter 1 (k) parameter
2	1	0	2 sample 1 (e) parameter
	2	0	2 sample 2 (e) parameter
		1	2 sample 1 (e) parameter 1 (k) parameter

NOTES:

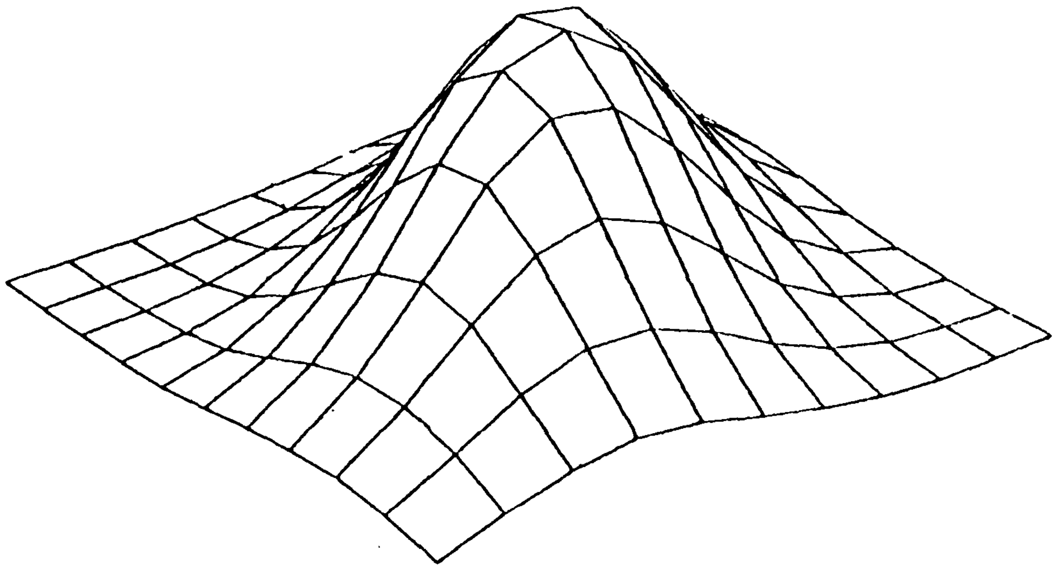
(e) : estimated by the user

(k) : known and provided by the user

to estimate the standard deviation of the given sample. The functional relationship he selects to estimate the standard deviation may or may not be the MLE for σ.

Output Interpretation. Figures 19 through 21 are examples of computer graphics generated during the estimation of an MLE for the standard deviation of a normal distribution with known mean. The random samples are given as 3 and 5. The correlation of random variables is stated to be 0.5. The mean is 3 and standard deviation is estimated to be 8 by the user. Figure 19, the first graphics screen, shows the shape of the PDF of the random variables. X ranges from -13 to 19 and y ranges from -11 to 21. The joint PDF of the random variable using the sample values 3 and 5, is computed as 0.002754317 when $\mu = 3$, $\sigma = 8$ and $\rho = 0.5$. Figure 20, the second screen, shows the likelihood function for the range of +/-2 centered around estimated standard deviation. Since the likelihood function continues to increase as the σ value decreases, a third screen is required to finally observe the MLE for σ. Figure 21 shows that when σ is 1.632993, the joint PDF is 0.02535282. This is the maximum value for the joint PDF of the given sample values 3 and 5, with $\sigma = 1.632993$ and $\mu = 3$. The difference between the two joint PDF values for sigma estimated by the user and for the MLE of σ is quite large, implying that the user's estimate for σ is far from the MLE for σ.

PDF for Given Mu : 3 Estimated S : 8



Y(L) [-11 21] Dy = 3.2 X(R) [-13 19] Dx = 3.2

When X is 3 Y is 5 PDF is 0.002754317

If you want to see the MLE for Sigma, range S+/-2,
hit Return when GO? appears. If not, hit the Break key

Fig. 19. Shape of PDF for the Normal Variables

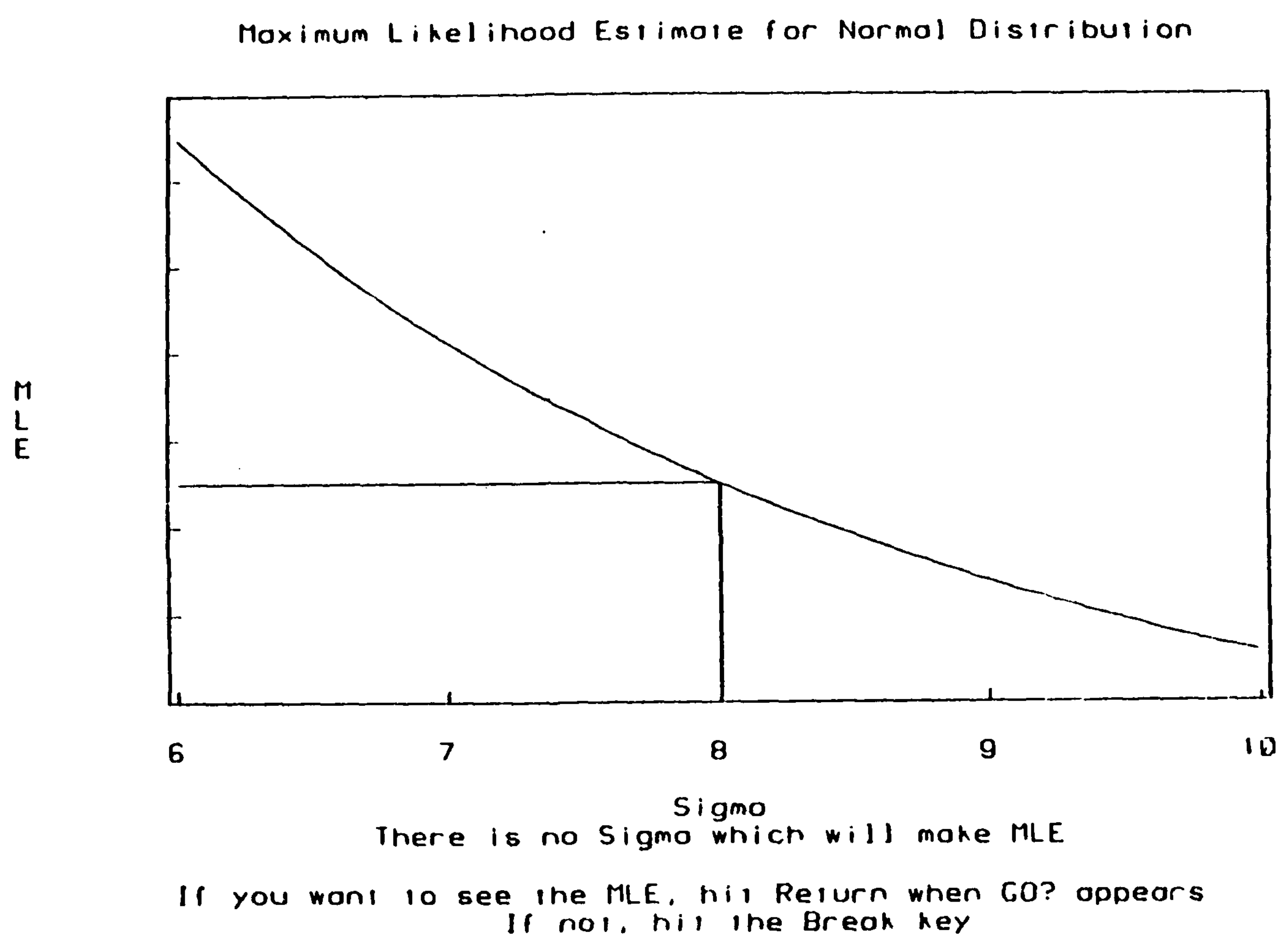


Fig. 20. Likelihood Function for Normal Variables

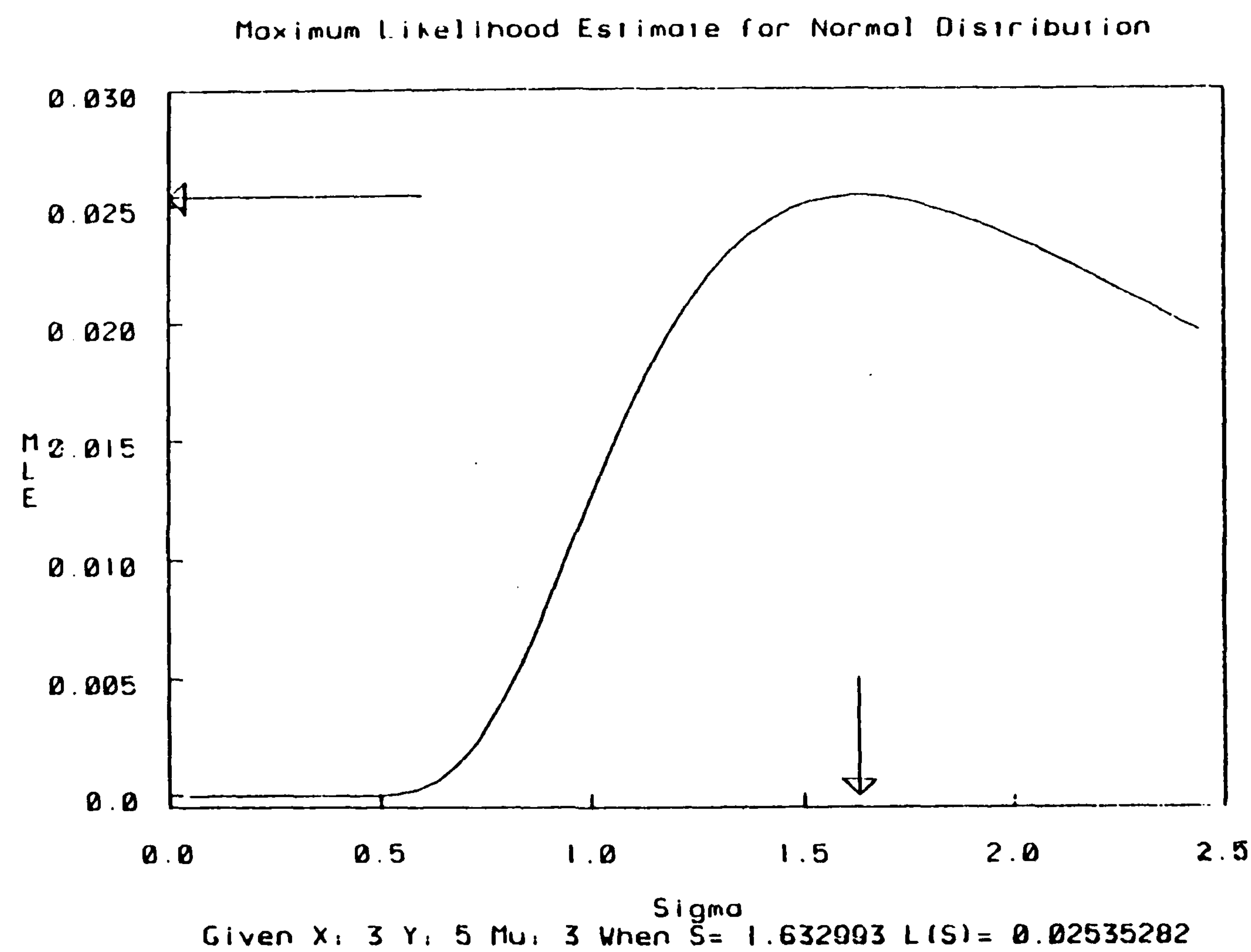


Fig. 21. Maximum Likelihood Estimation for Normal Variables

Another example of maximum likelihood estimation is given using the standard gamma distribution. Here, the MLE for α, which is a parameter of the gamma distribution, is not unique or is not computable (6:99; 9:229). Thus computer graphics must be used to show iterations needed to discover the MLE for α. Since there is no closed form solution which allows the MLE for α to be computed directly, different α values are put into the joint PDF in search of a particular α that maximizes the joint PDF. Since only a subset of the infinite number of possible values could be selected, one can never be sure that any particular α value maximizes the joint PDF through the whole range of α. However, in practice, the true maximum likelihood estimate for alpha will exist close to some selected α value. Therefore, at each succeeding iteration, a narrower range for α, centered at the previously selected α value, will be used to compute the joint PDF. Through repeated iterations, an approximation for the functional form of the maximum likelihood estimator for the gamma parameter can be achieved.

An example is given using the standard gamma distribution whose sample value is 5 and whose estimated α is given as 2. Figure 22, the first screen, shows that the PDF for the sample value 5, when α is 2, is 0.03368973. Figure 23, the second screen, shows the maximum likelihood estimation for α when α values ranging from 0 to 10 are

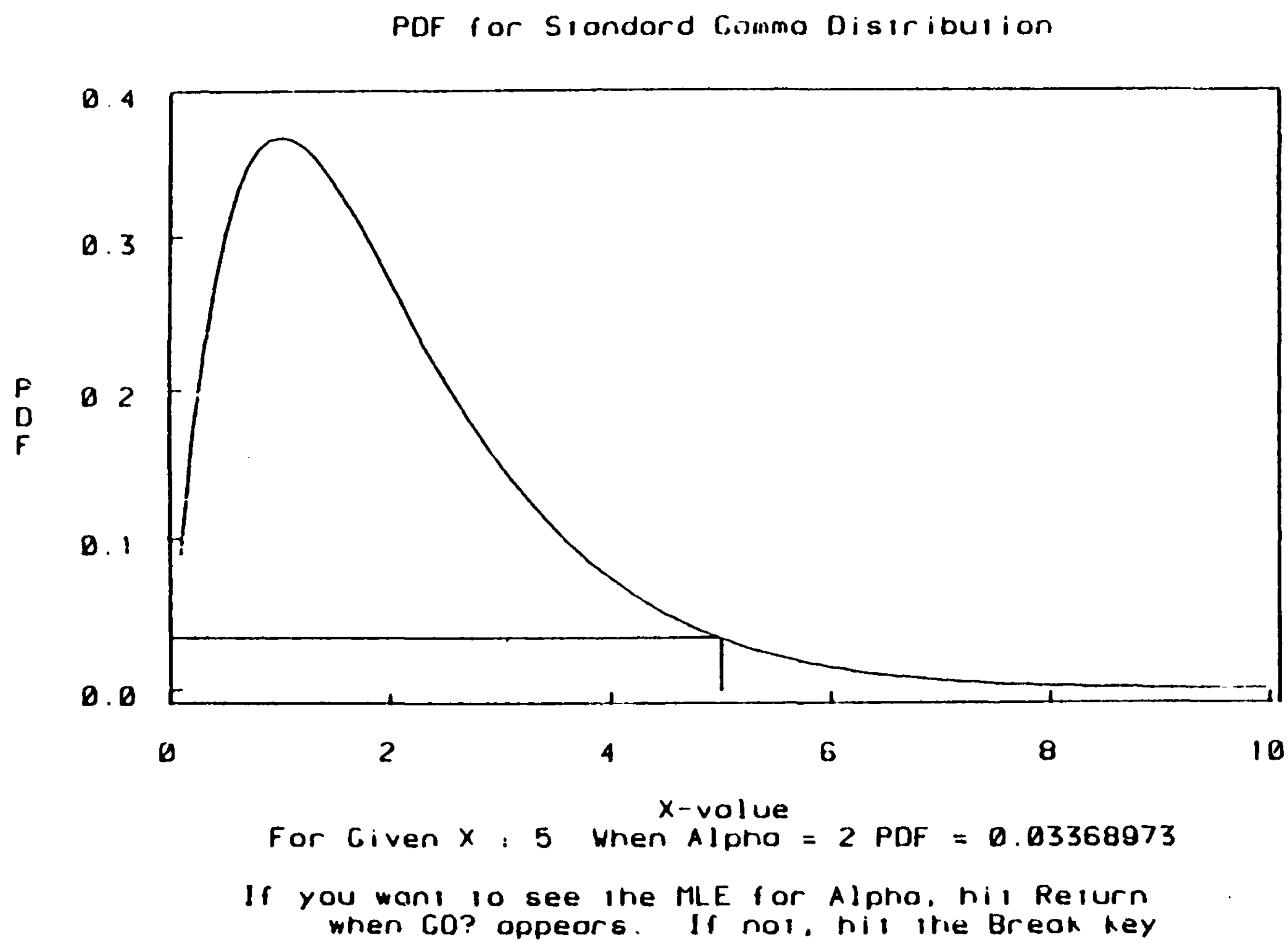


Fig. 22. Shape of PDF for the Gamma Variables

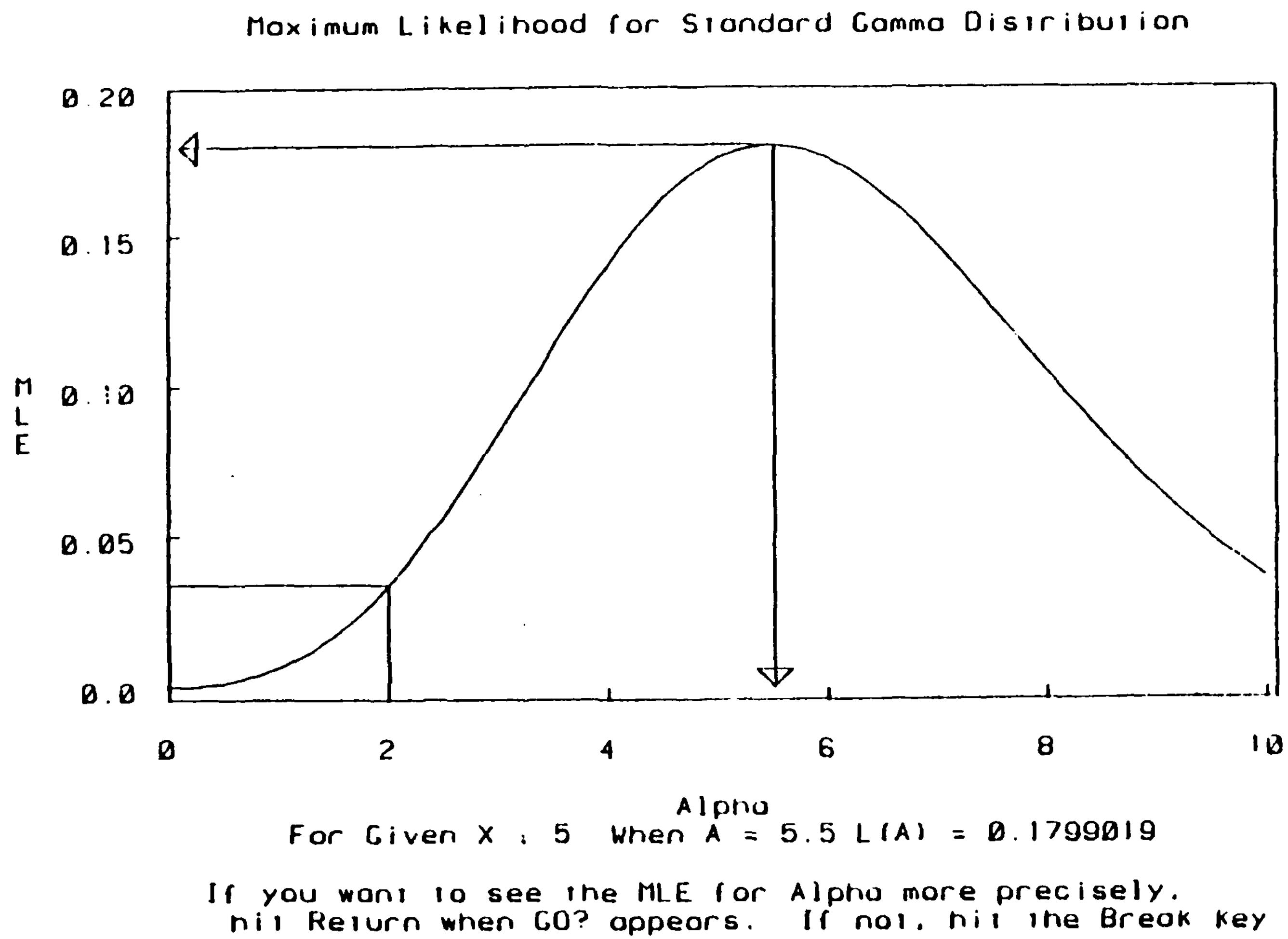


Fig. 23. Maximum Likelihood Estimation for the Gamma Variables

used to compute the PDF for the sample value 5. At the first iteration, the MLE for α is estimated as 5.5 and the PDF is 0.1799019. Figure 24, the third screen, uses a narrower range of α from 5 to 6. The second iteration results in an MLE equal to 5.49 and a PDF value of 0.179903. This is larger than the previous PDF by .0000011. Figure 25, the fourth screen, uses an even narrower range for α and shows the same value for the MLE and PDF as Figure 24. However, four vertical arrows appear on the screen, which signifies that the MLE may not be unique nor analytically computable. Figure 26, the fifth screen, shows that the PDF plot is not a smooth line any more and the MLE for α is 5.49072 and the PDF is 0.1799031. But the α value is not unique. After four iterations, it can be concluded that α is not unique but exists and is approximately 5.49.

Transformation of Random Variables

As Table 9 (page 70) points out, the user must access the ISTP transformation of random variables learning module by menu selection. Figure 3 (page 61) indicated that the highest macro in the transformation of random variables is ?tran and that lower level macros are: ?uvtran, for the univariate case; ?bvtran, for the bivariate case; ?trdis, for the univariate discrete case; and ?trcont, for the univariate continuous case. Table 7 (pages 51-53)

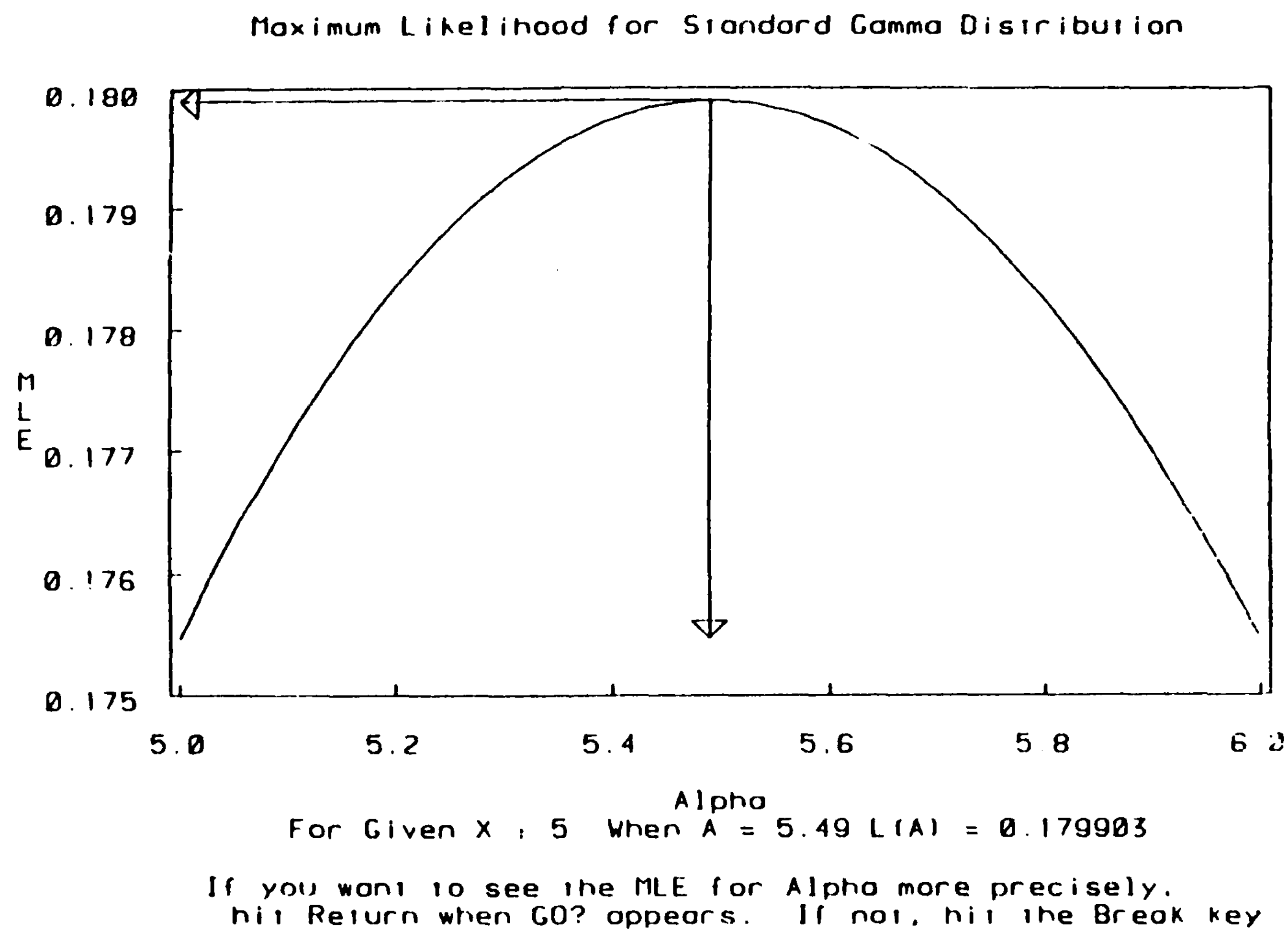


Fig. 24. Iterated Process of Maximum Likelihood Estimation (2nd Iteration)

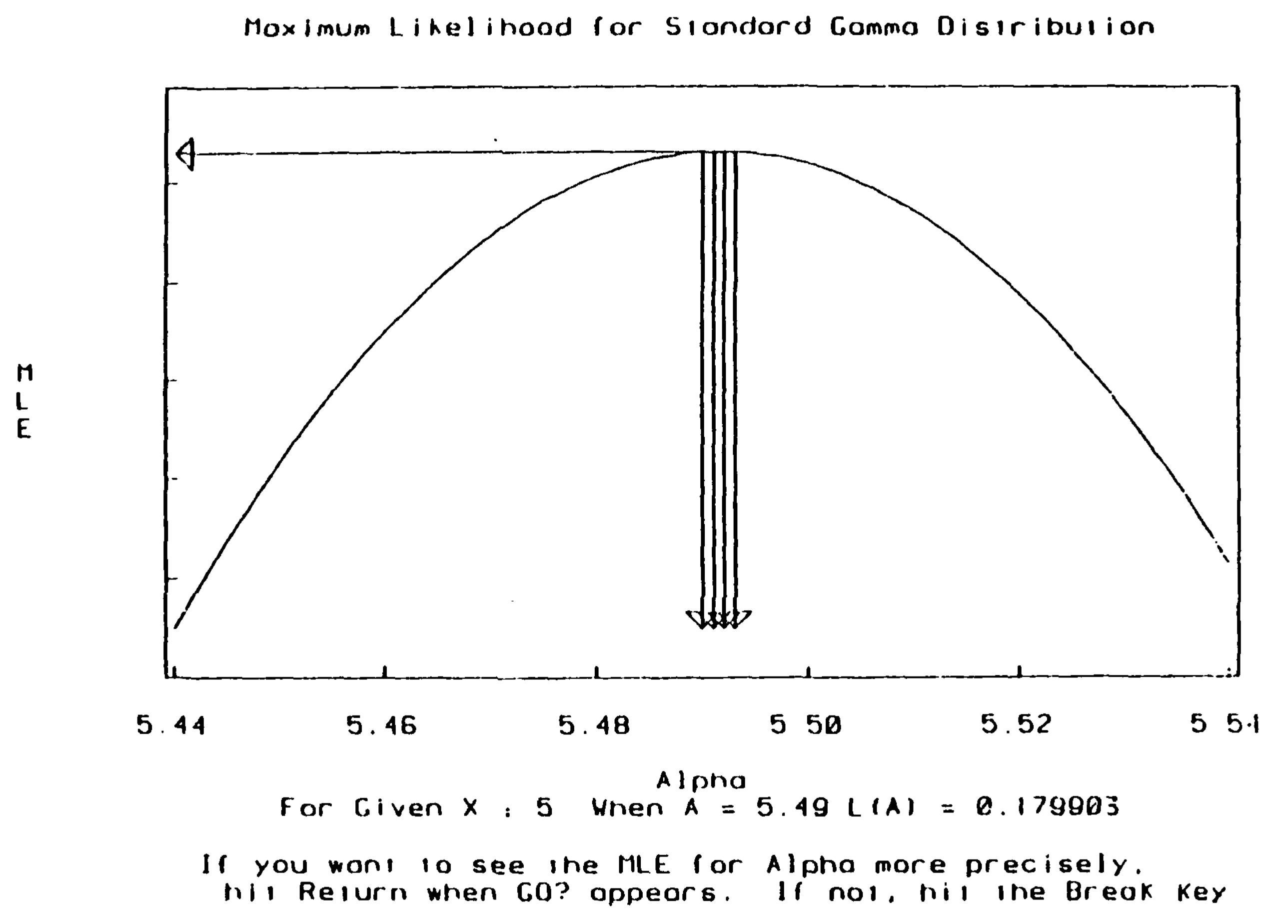


Fig. 25. Iterated Process of Maximum Likelihood Estimation (3rd Iteration)

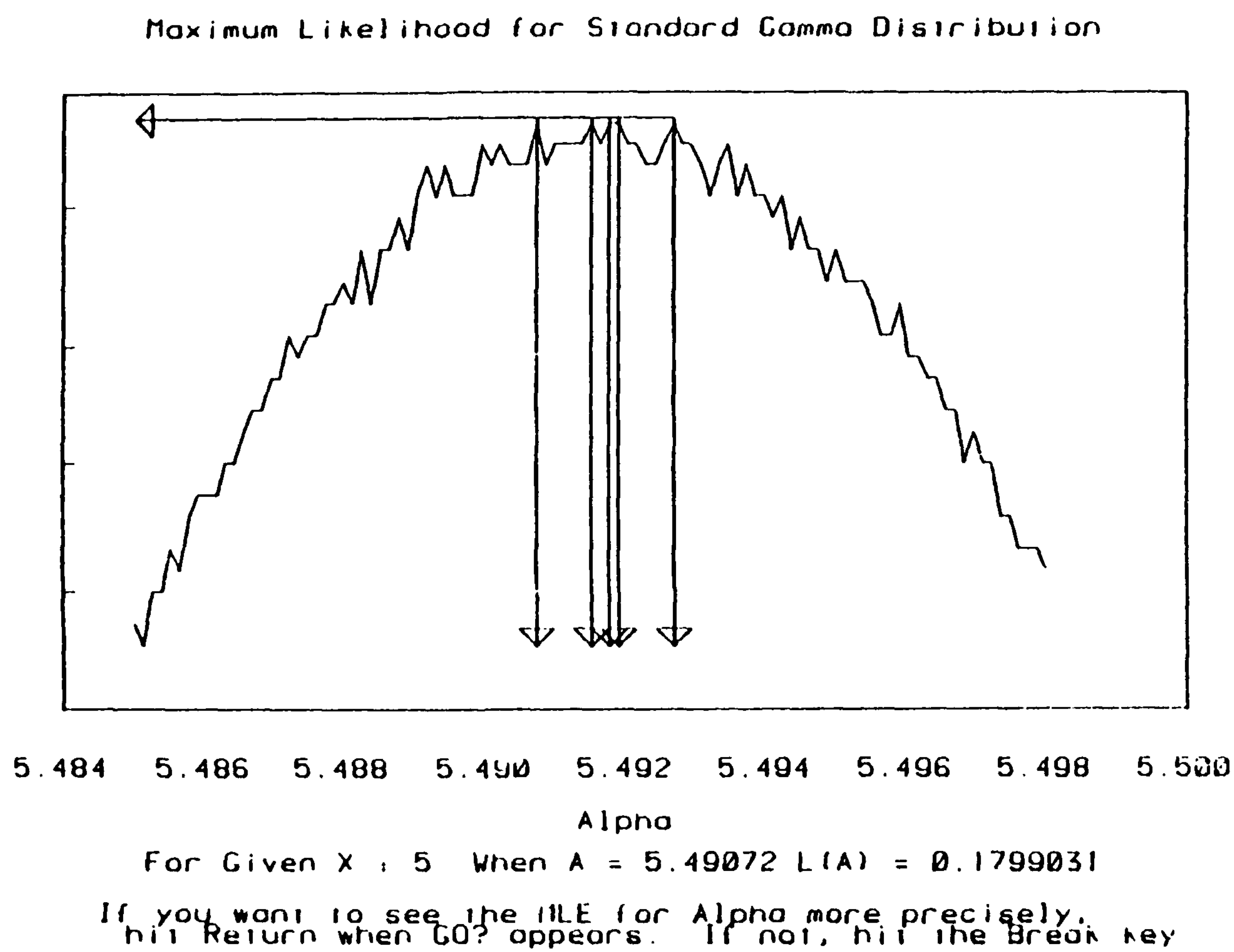


Fig. 26. Iterated Process of Maximum Likelihood Estimation (4th Iteration)

provides a list of the lowest level of macros and available transformation forms for univariate discrete and continuous random variables and bivariate continuous random variables.

Input Requirements. As discussed in Chapter III, transformations are accomplished by a known PDF to an estimated PDF transformation. The accuracy of the estimated PDF depends on the width of the integration subinterval that the user requests. The narrower the subinterval, the more precise the outcome. However, if too many intervals are required, the length of the time it takes the computer to perform the transformation may become unacceptable. On the other hand, if too few intervals are requested, the transformation may be inaccurate. The decision concerning how many intervals to compute depends on at least the following factors: the user's accuracy and precision needs, the amount of computer time available for such computation, and the number of users on the computer at the time the transformation is solicited.

Depending upon the type of transformation requested, there may be certain restrictions placed on the range of the values of random variables involved. For certain complicated transformations, the range of transformed variables is estimated by the ISTP to avoid generation of meaningless displays.

Univariate Discrete Case. The transformation of a binomial random variable is the only transformation available for the discrete case. Since the transformation is for the polynomial $Y = aX^2 + b \cdot X + c$, where X is a binomial random variable and Y is the transformed random variable, the random variable X is paired with the transformed variable Y discretely. A given random variable Y may be associated with one or two values of the random variable X. If one value of X is paired with one value of Y, the PMF at X is the PMF at Y. On the other hand, if there are two values of X which are paired with one value of Y, the sum of probabilities of two values of Y becomes the probability associated with the value of Y.

Univariate Continuous Case. Functions of random variables for which a transformation can be made are: $Y = X^2$, $Y = \mathrm{Log}(X)$, $Y = \mathrm{Exp}(X)$, $Y = \sqrt{X}$ and $Y = a\,X + b$, where X is the original random variable and Y is the transformed random variable. The inputs required to transform univariate continuous random variables are: the parameter(s) of the distribution from which the original random variable comes, the smallest value of Y and the largest value of Y for which the PDF of Y is desired, and the sub-interval of Y. Caution should be exercised when the user selects the range of Y. Y may have restrictions on its range because of restrictions on X and/or the form of its

transformation equation. The explicit restrictions on Y's range for the available transformations are summarized in Table 16.

It is sometimes difficult to compute the appropriate range for Y. A recommended way is to establish the range of Y in terms of the range of X, because the values of the random variable X are known _a priori_ while the values of the random variable Y are not known until the transformation has been performed. For example, to set the range of Y for the transformation Y = Log(X) when X $\geq$ 0, instead of using specific values for the smallest and largest value of Y, the Y range can be given in terms of X; that is, Log(0.1) for the starting point of Y, and Log(4) for the ending point of Y.

Bivariate Continuous Case. The transformation of bivariate random variables are more involved than those of univariate random variables. Since it can be a very difficult task to establish the appropriate bounds on Z for the transformation Z = f(X,Y), the ISTP automatically computes the best range for Z. However, the user is afforded the opportunity to adjust the number of suggested sub-intervals. By asking for a large number of intervals, the user can achieve a high level of accuracy. By requesting a small number of intervals he can obtain speedy results. In all cases the user must provide the

TABLE 16

RESTRICTIONS ON Y RANGE FOR TRANSFORMATION OF RANDOM VARIABLES

Transformation Equation	Defined X Range	Restrictions on Y Range (or X)
$Y = X^2$	$-\infty \le X \le \infty$	$Y \ge 0$
	$X \ge 0$	
$Y = \text{Log}(X)$	$-\infty \le X \le \infty$	$X > 0$
	$X \ge 0$	
$Y = \text{Exp}(X)$	$-\infty \le X \le \infty$	$Y > 0$
	$X \ge 0$	$Y \ge 1$
$Y = \sqrt{X}$	$-\infty \le X \le \infty$	$Y \ge 0,\ X \ge 0$
	$X \ge 0$	
$Y = a X + b$ $(a \ne 0)$	$-\infty \le X \le \infty$	No restriction
	$X \ge 0$	$(Y-b) \cdot a \ge 0$

parameters of distributions of the random variables involved in the transformation.

Output Interpretation. The steps involved with transformation of a log-normally distributed random variable X to Y, Y = Log(X), are illustrated by Figures 27 through 30. These represent the sequence of computer graphics produced by the ISTP learning module for the transformation of a random variable. The inputs for this transformation are: the mean and standard deviation of the log-normal distribution, 0 and 1, the starting and ending values for Y, -2 and 2 and the sub-interval for Y, 0.05. Figure 27, the first graph, shows the relationship between random variables X and Y and between Y and X. Figure 28, the second graph, shows the relationship between the distribution of X and the distribution of Y which turns out to be the relationship between a lognormal distribution and a normal distribution. This graph shows that Y is distributed normally with mean 0 and standard deviation 1.

Figures 29 and 30 display the iterations of the transformation process. Figure 29 represents the transformation of the first interval, and Figure 30 represents that for the 45th interval. In Figure 29, the shaded areas under the PDFs of X and Y represent the same probabilities. Dividing the shaded area of Y by the increment 0.05 yields the value of Y's PDF. In Figure 29, the PDF of Y is 0.0567585 when Y equals -2. Utilizing the same process,

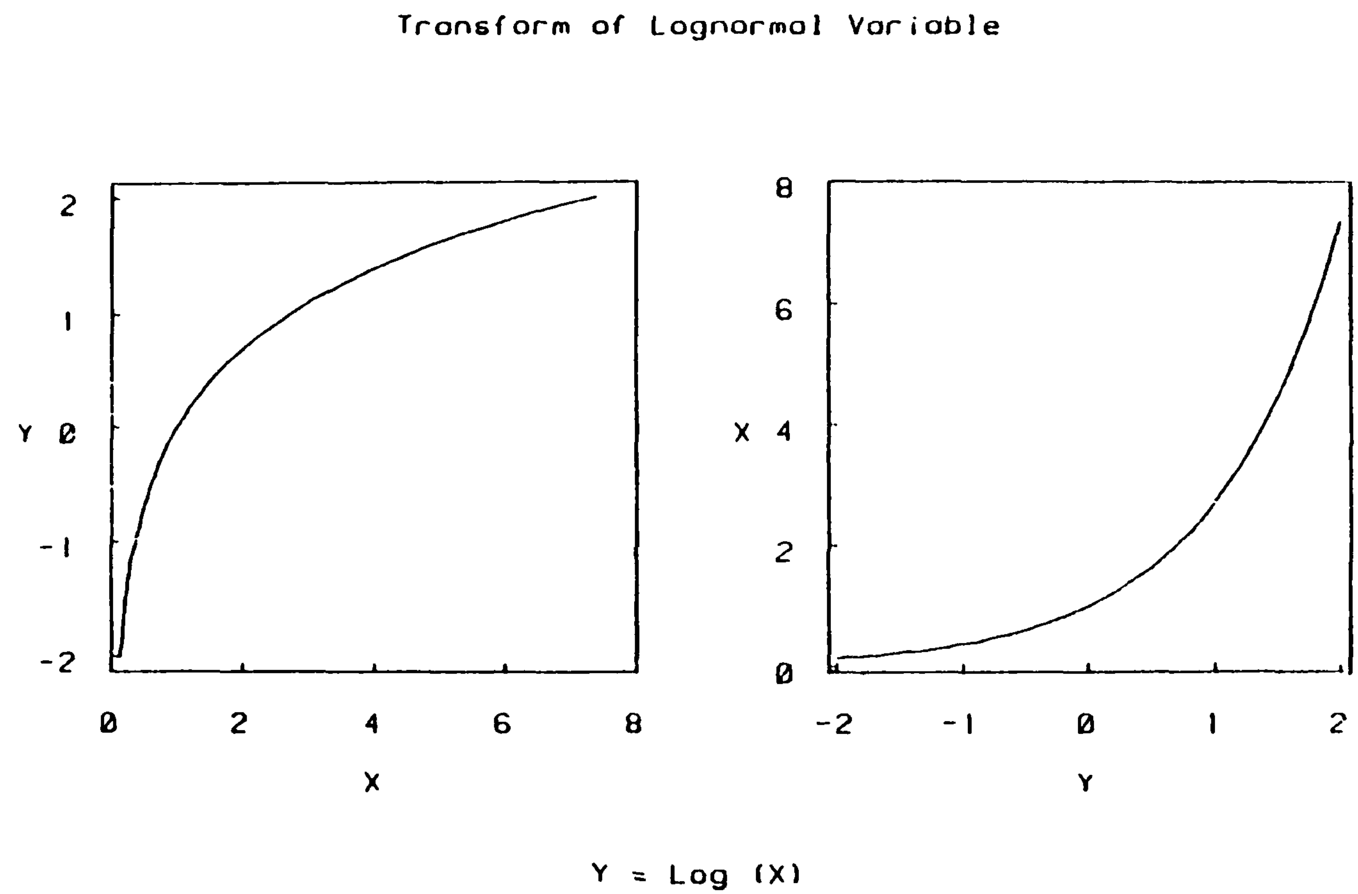


Fig. 27. Relationship between X and Y when Y = Log(X)

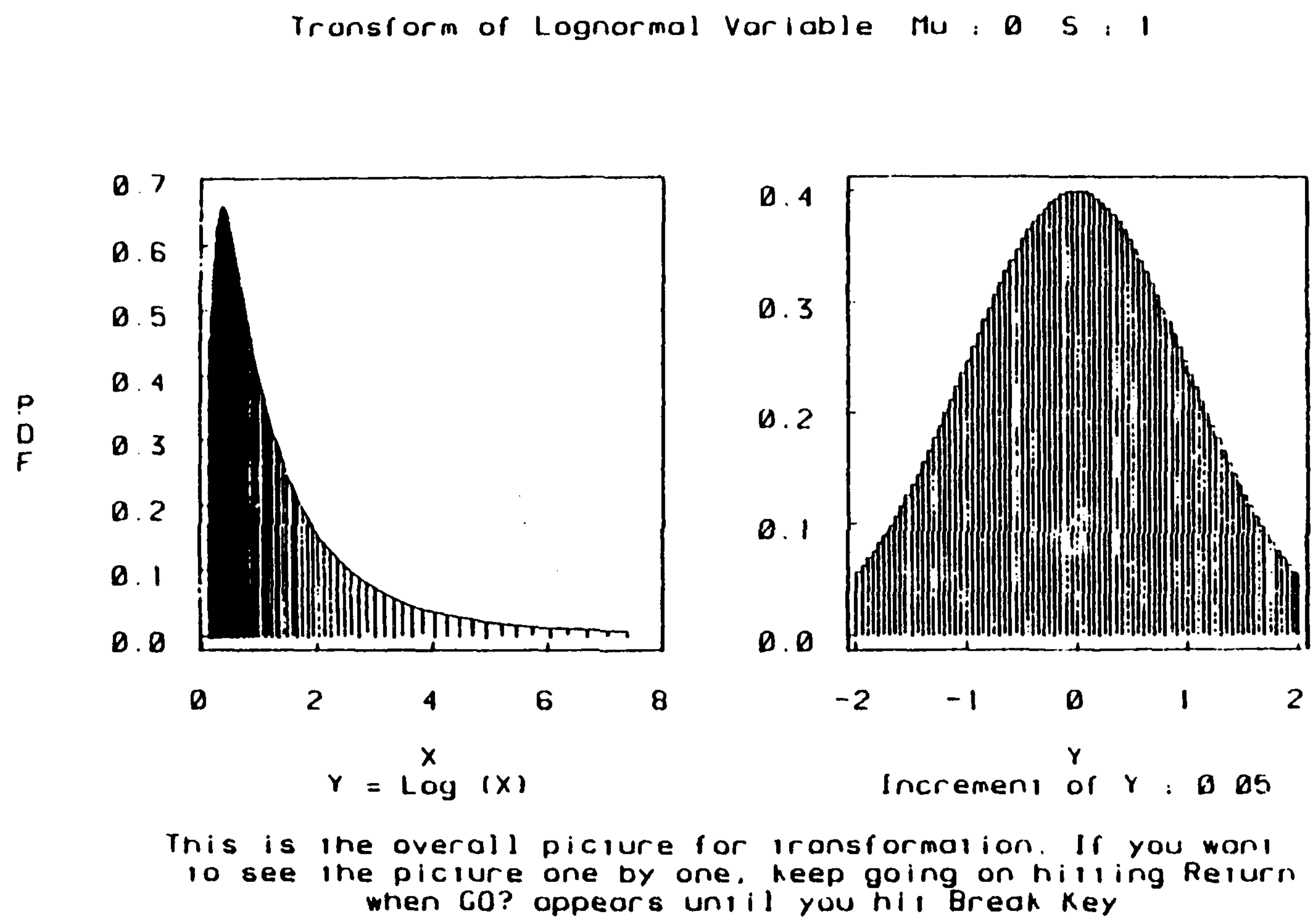


Fig. 28. Relationship between PDF of X and PDF of Y
when Y = Log(X)

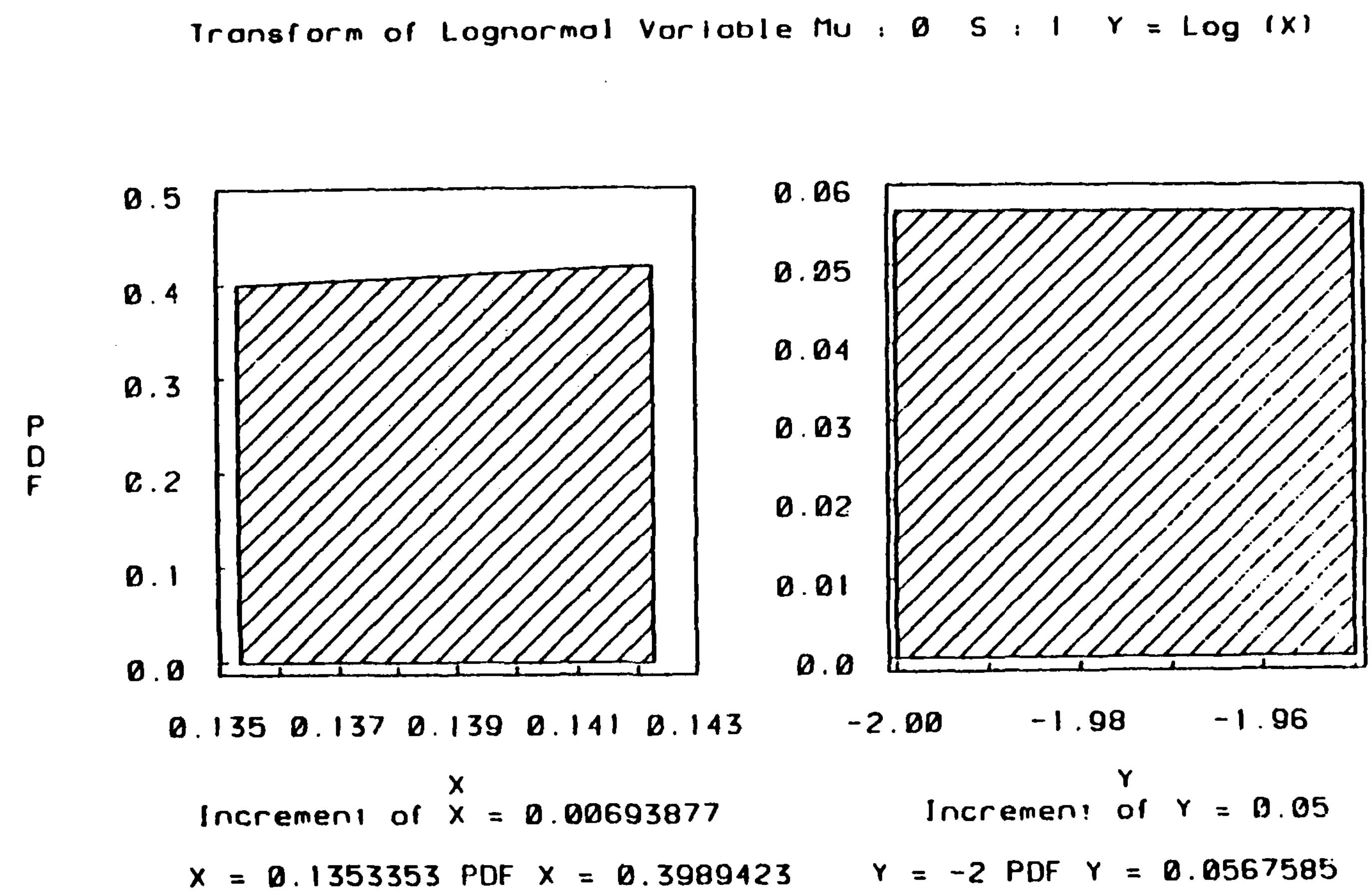


Fig. 29. Iterated Process of Transformation (1st Iteration)

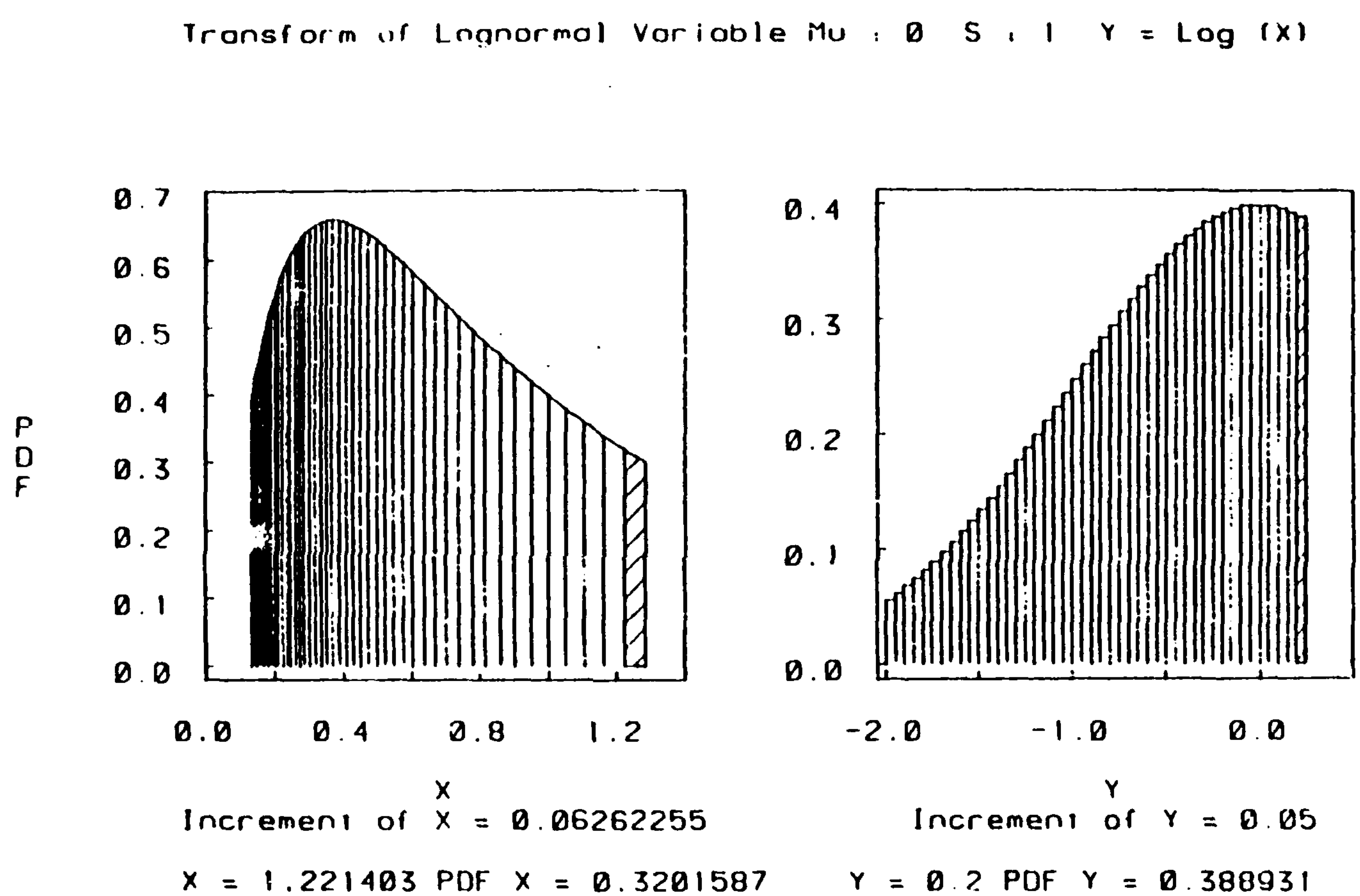


Fig. 30. Iterated Process of Transformation (45th Iteration)

Figure 30 indicates the PDF's value when Y equals 0.2 is found to be 0.388931.

The derivation of the test statistic T as the ratio of a standard normal random variable and the square root of a chi-square random variable divided by its degree of freedom is another important transformation that the ISTP can carry out. Figure 31 displays the first screen provided by the ISTP transformation routine. It contains two graphs. The one on the left illustrates the relation between T and $\frac{X}{Y}$, where Y is defined as $\sqrt{Yp/n}$, X~N(0,1) and Yp~Chisq(n). The shaded area on the right graph represents the region of integration over the X and Y plane. Figure 32, the second screen, also contains two graphic images--a contour plot for the PDF of the bivariate original random variables, and a three-dimensional plot for the same PDF. Figure 33, the third screen, provides the transformed PDF with the original PDF. The total area encompassed by the PDF for the transformed variable is 0.998133. This value means the CDF for a T distributed random variable over the range -4 to 4 is 0.998133. This is very close to the exact cumulative probability for T with 11 degrees of freedom from -4 to 4, which is 0.997914. The slight inflation results from numeric approximation of the integration via the routine of the IMSL library. Intervals of the transformed variable and the original variable are matched on a one-to-one basis. This

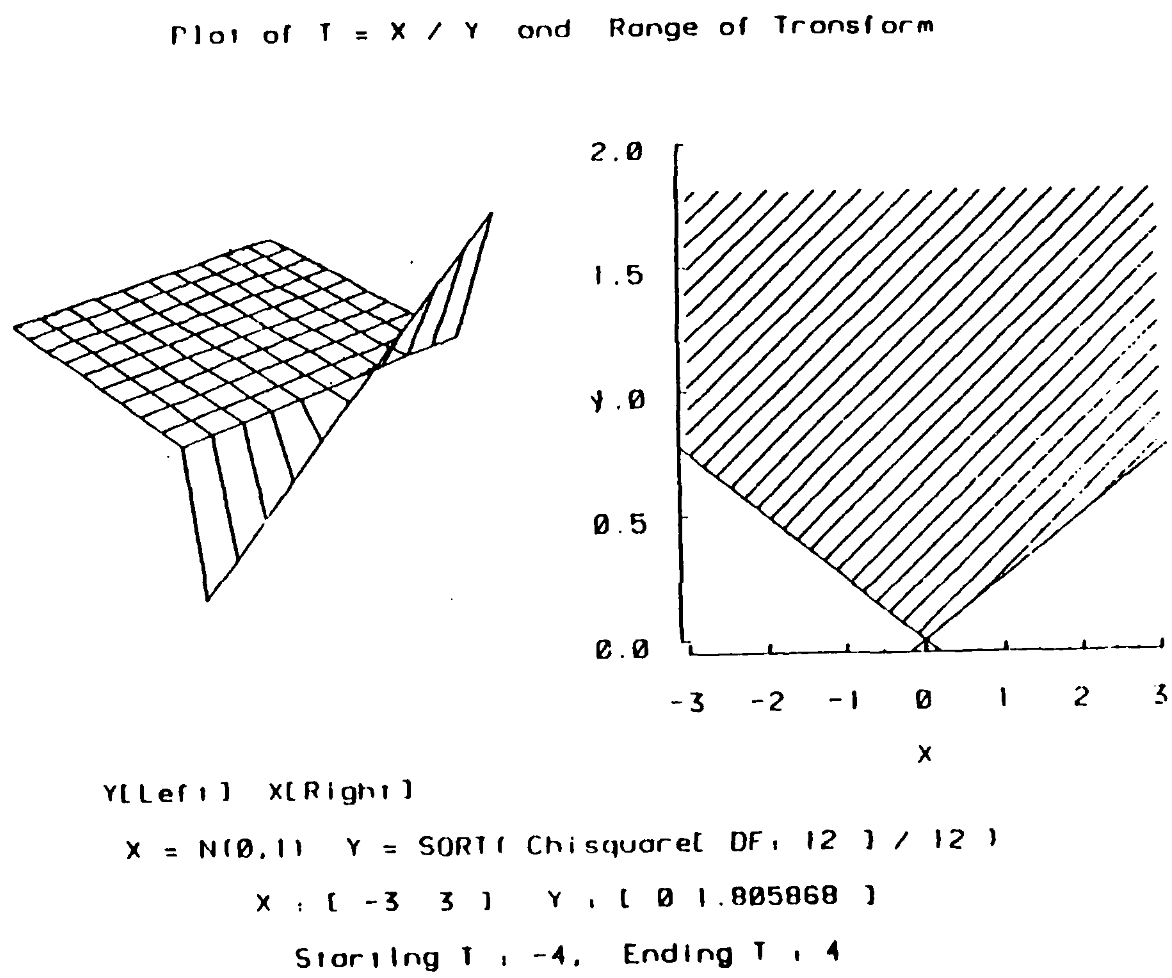


Fig. 31. Relationship between T and X/Y and the Defined T Range

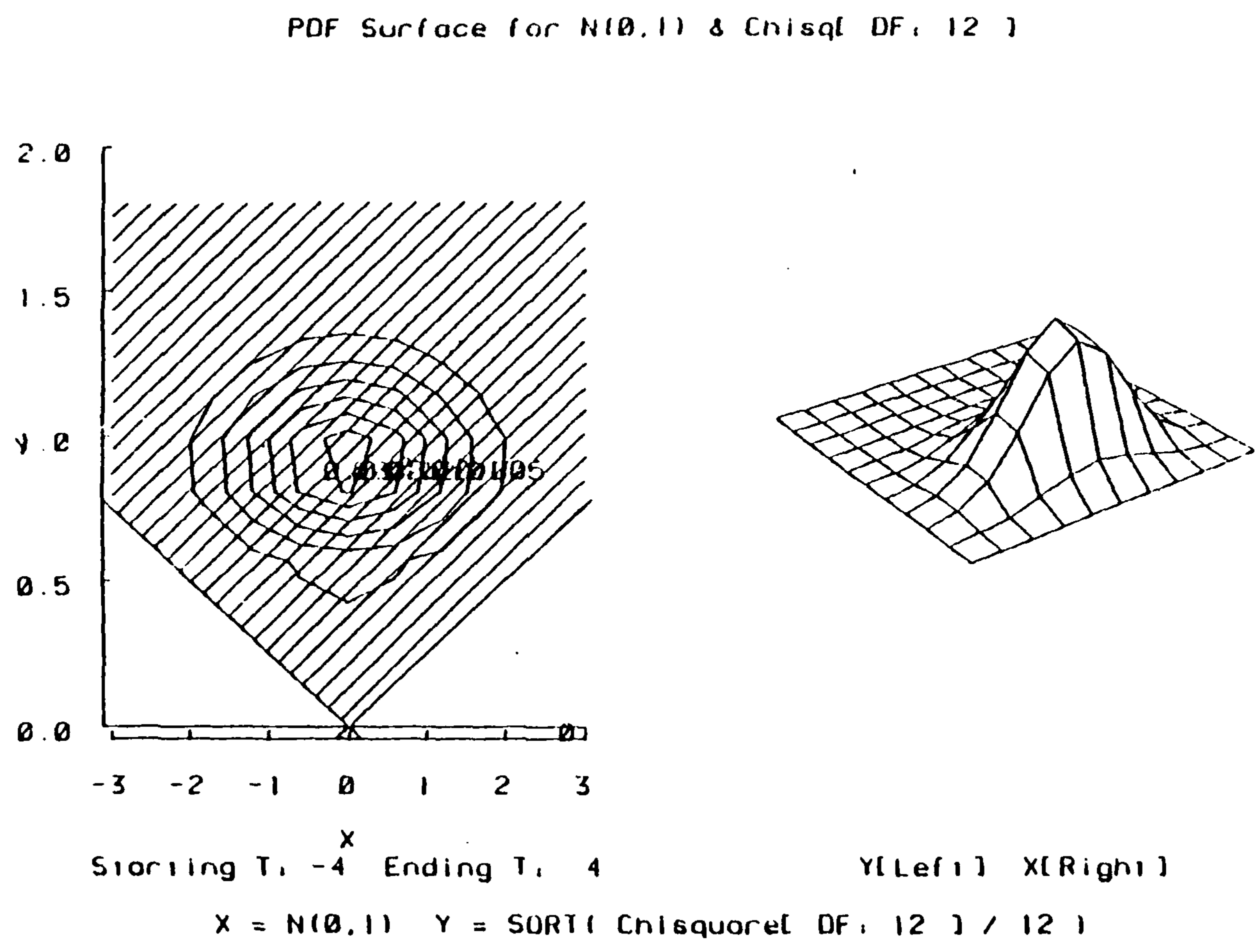


Fig. 32. Shape of the PDF for the Original Distribution

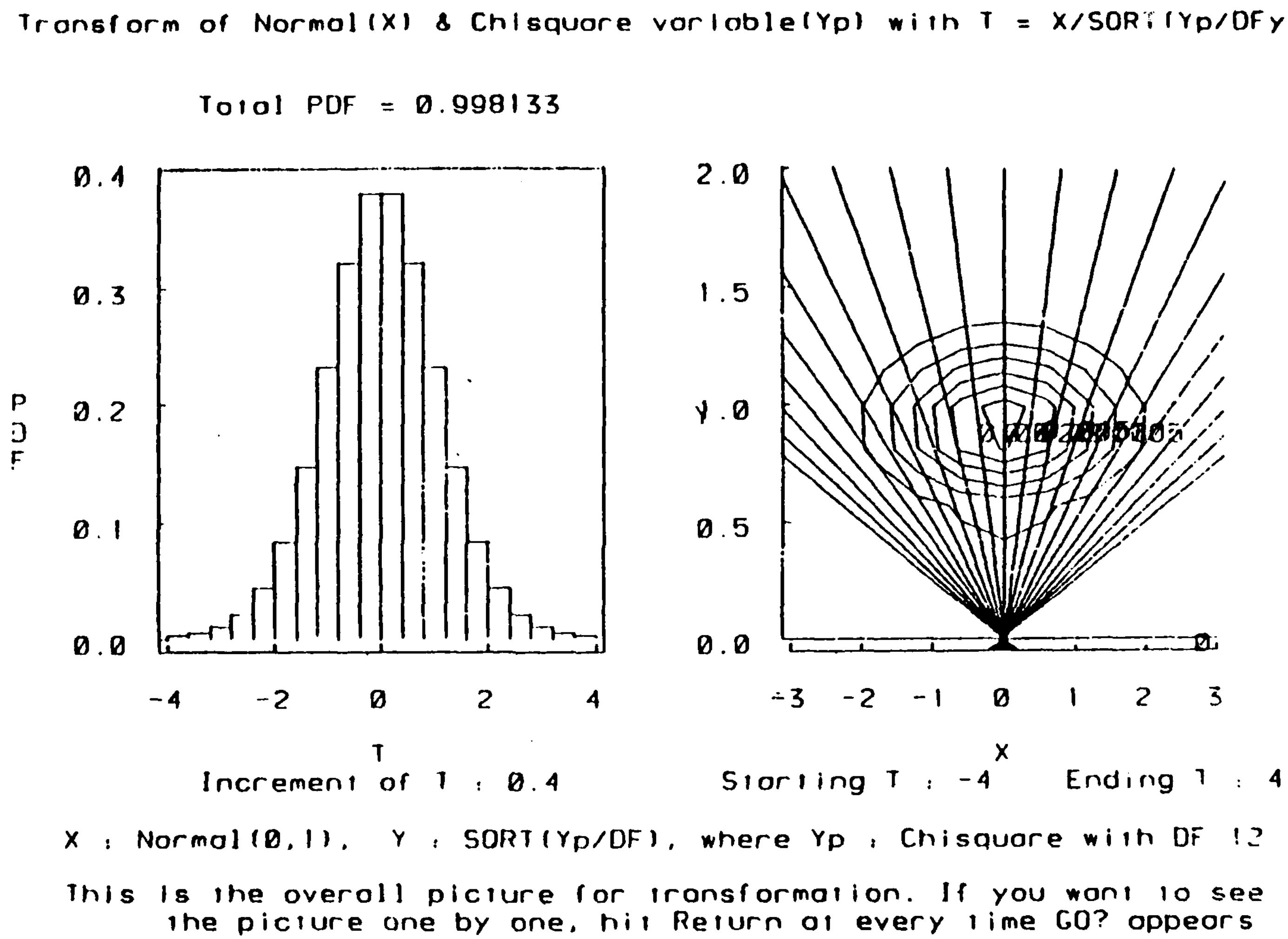


Fig. 33. Relationship between Transformed PDF and Original PDF

means that the volume under each three-dimensional slice over the region of integration defined by a subset of values for X and Y is transformed into the area under the transformed PDF over the range of T (-4,4) set by the ISTP program. Figures 34 and 35 show the 1st and 15th iteration of this transformation.

Hypothesis Testing

Figure 3 (page 61) indicated that the macro ?hypo is the highest macro in the hypothesis testing area and that ?test1 and ?test2 can be used to invoke a test for the mean of a normal distribution when the standard deviation is known and when the standard deviation is unknown. Table 17 provides the information concerning input requirements for these two tests.

Input Requirements. Five inputs are required to use either ?test1 or ?test2: the sample mean $\overline{X}$, the sample standard deviation S, the sample size n, the null value μ_o, type I error σ, and the effect size.

Effect size is "an index of degree of departure from the null hypothesis" (7:10). The relation between effect size and power and between the effect size and the sample size is as follows:

> The larger the effect size posited, other things (significance criterion, sample size) being equal, the greater the power of the test. . . . The larger the effect size posited, other things (significance criterion, desired power) being equal, the smaller the sample size necessary to detect it. (7:11)

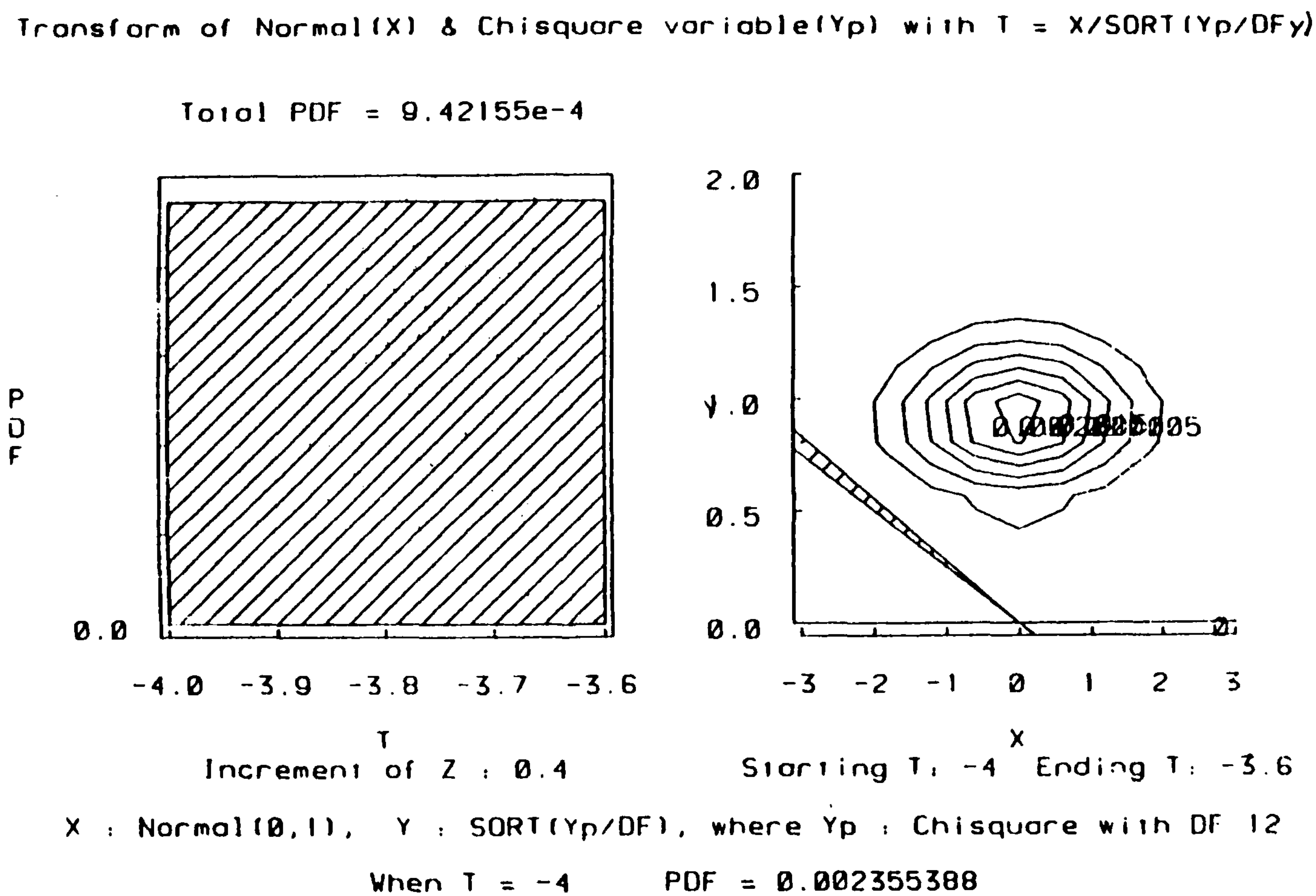


Fig. 34. Iterated Process of Transformation (1st Iteration)

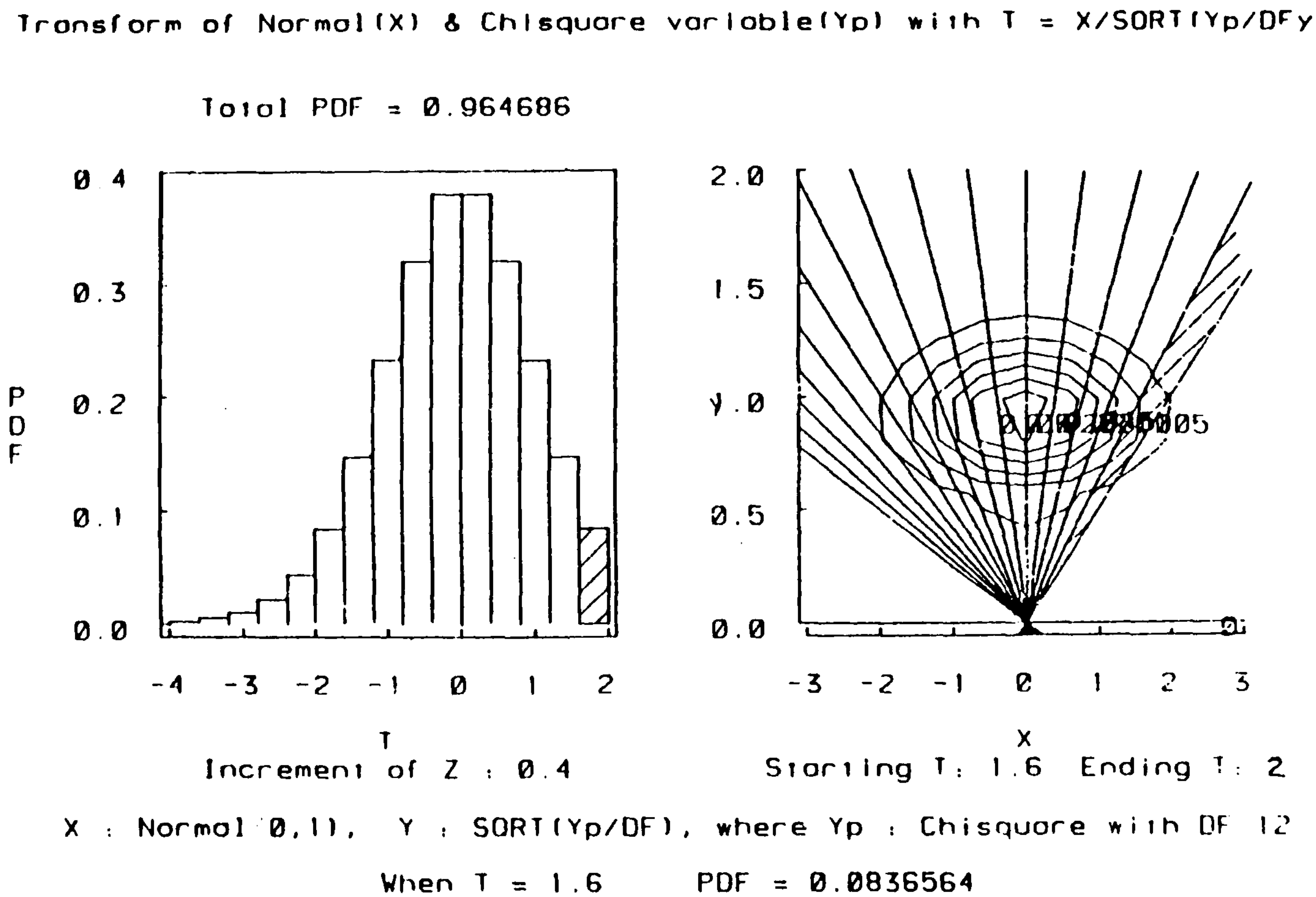


Fig. 35. Iterated Process of Transformation (15th Iteration)

TABLE 17

DESCRIPTIONS OF HYPOTHESIS TESTING MACROS IN THE ISTP

Types	Macros	Descriptions	Inputs
Known σ	?test1r(x,n,s,m,a,e)	right tail test	x : sample mean ($\bar{x}$) n : sample size s : sample standard devia. m : μo a : type I error (α) e : effect size
	?test1l(x,n,s,m,a,e)	left tail test	
	?test1t(x,n,s,m,a,e)	two tail test	
Unknown σ	?test2r(x,n,s,m,a,e)	right tail test	
	?test2l(x,n,s,m,a,e)	left tail test	
	?test2t(x,n,s,m,a,e)	two tail test	

NOTE: Test for the Mean of Normal Distribution.

In the ISTP, the effect size represents the absolute distance between the null value μ_o and a presumed alternative value for the parameter μ_1. Therefore, the power and the type II error are computed based on size of the effect value. To avoid confusion, the user is required to provide the effect size in terms of deviation without considering its direction. For example, for a right tail test, the alternative value μ_1 will be greater than μ_o, and for the left tail test, μ_1 will be smaller than μ_o. However, when providing the effect size, the user need not consider this condition. Instead, the ISTP will determine the direction once the value of effect size is given by the user. To keep the graphics simple, only the right alternative μ_1 value is displayed for the two tail test.

<u>Output Interpretation</u>. An example is given for the case of the test of a mean of a normally distributed random variable when σ is unknown. $\overline{X}$, the sample size and the sample standard deviation, are given as 12.5, 15 and 2.3. μ_o is given as 11.0. The desired level of type I error is presumed to be 0.05 and the effect size is 1 for this right tail test. Figures 36 through 38 display the computer graphics produced by the ISTP.

Figure 36, the first screen, shows the relationship between the populations assuming the null hypothesis is true (smooth line) and assuming the alternative hypothesis is true (star line). Although the standard deviation is

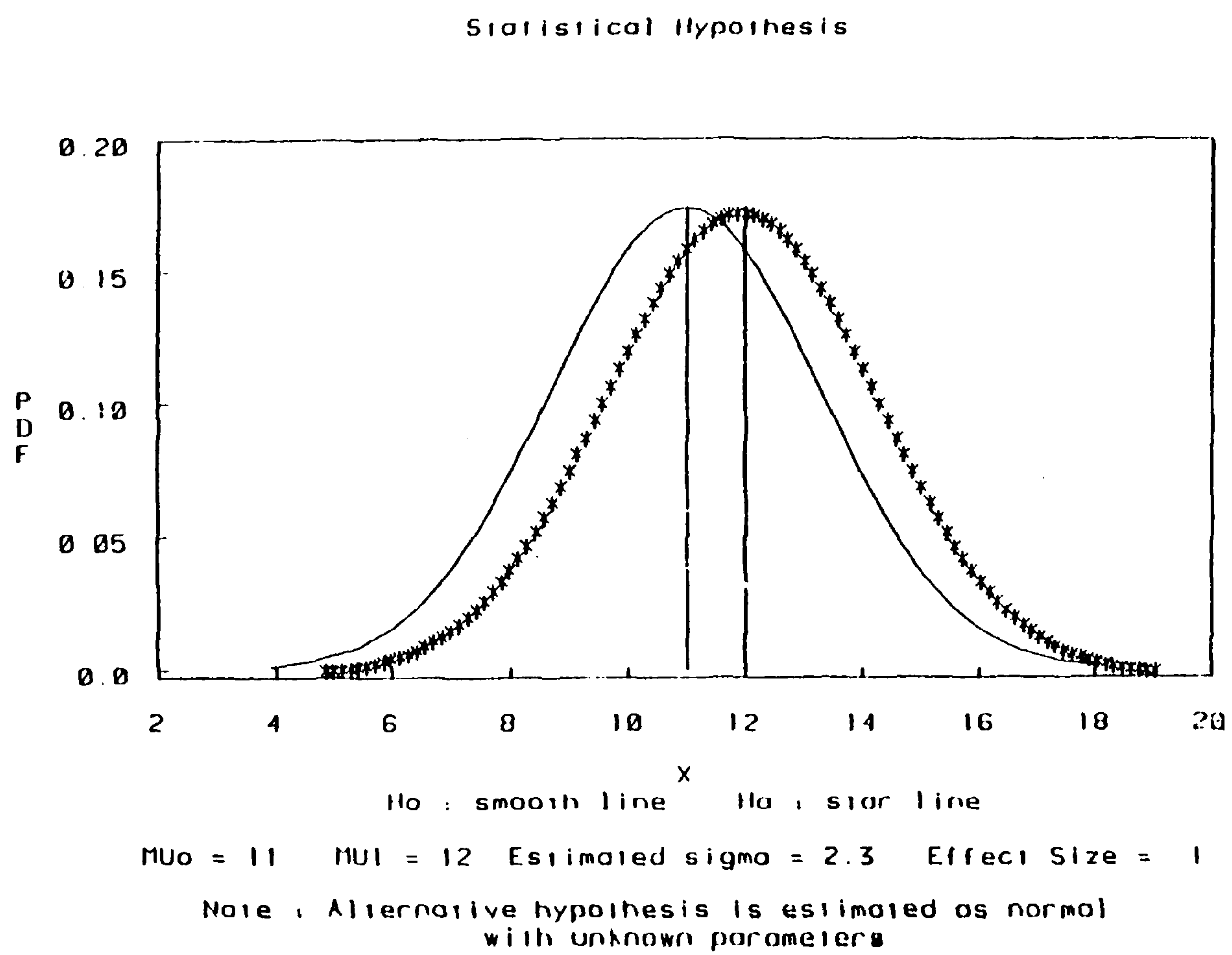


Fig. 36. The Statistical Hypothesis

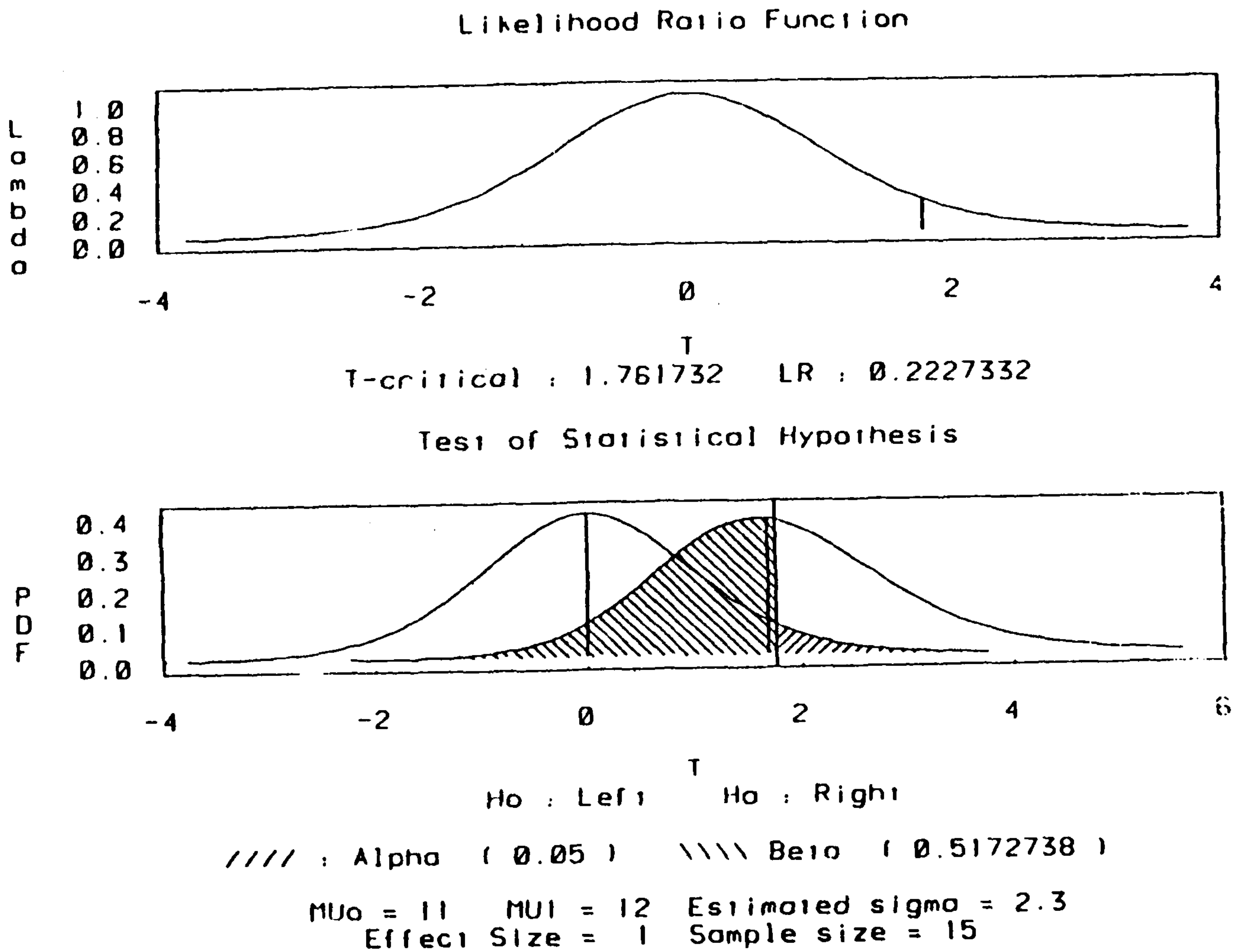


Fig. 37. Likelihood Ratio Function and Test of the Statistical Hypothesis

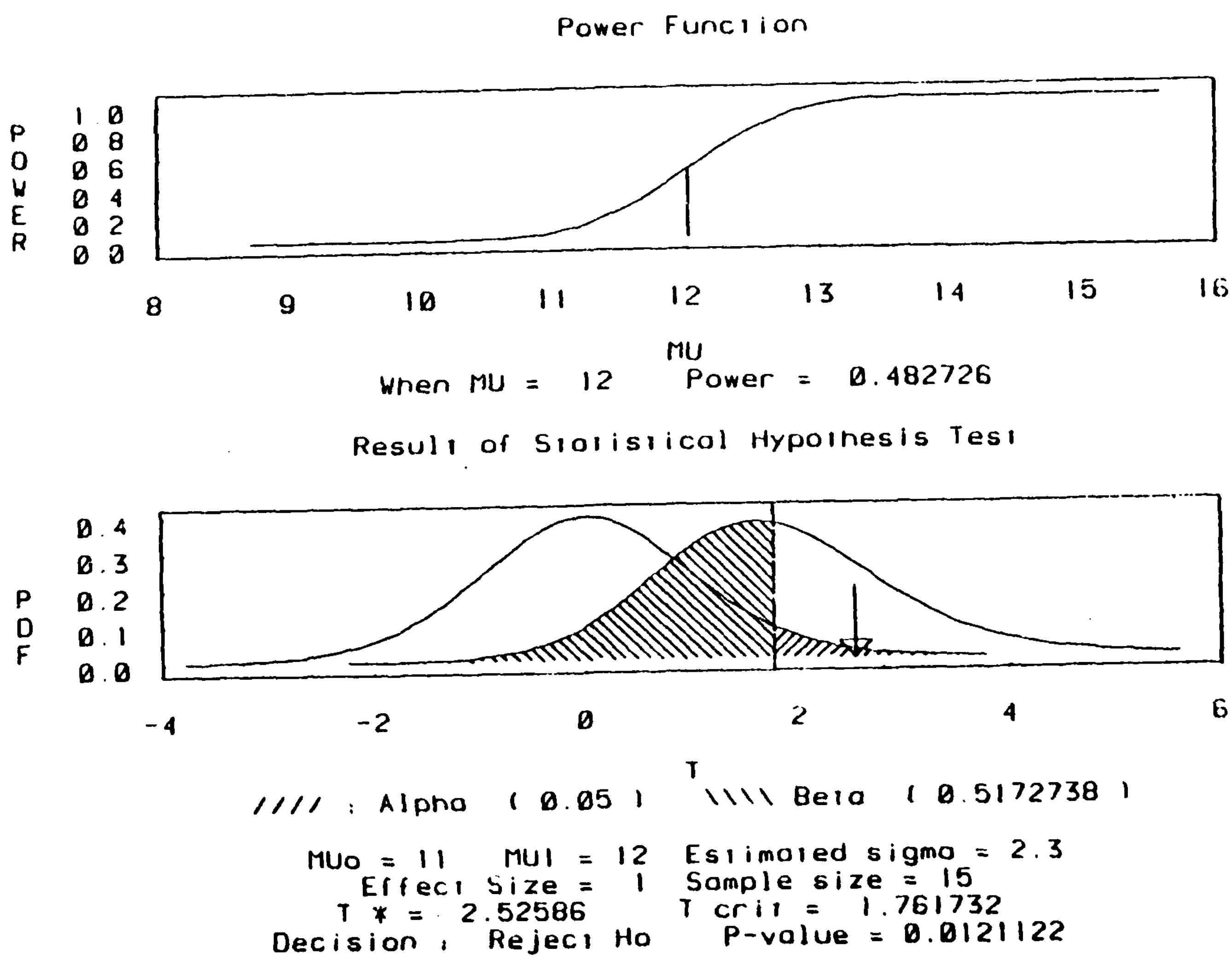


Fig. 38. Results of the Statistical Hypothesis Test

unknown, σ can be estimated by the sample standard deviation. The mean of the alternative population is 12 because the effect size is set to 1 and this is a right tail test.

Figure 37, the second screen, shows the likelihood ratio function and the T distribution under the null hypothesis and under the alternative hypothesis. The T critical value is 1.761732 and the likelihood ratio is 0.2227332 at the T critical value. The two T distributions show that the type I and type II errors are 0.05 and 0.5172738, respectively. The area representing type I error is to the right side of the critical value. It represents the probability of rejecting H_o when the H_o is true. The area for type II error, on the other hand, is to the left side of the critical value. This represents the probability of accepting H_o when H_o is false.

Figure 38, the third screen, shows the power function and the decision and final results of the test. The power of the test when the effect size is 1(μ = 12) is 0.482726, which is equal 1 - β. The computed value of the test statistic, T, falls to the right of the T critical value. Since the test statistic T (2.52586) is larger than the T critical value (1.761732), H_o must be rejected. The prob value is 0.0121122. It can be compared directly to the type I error. If the prob value is larger than the type I error, H_o should be rejected.

Power Function. For a Z test, the power can be computed using the Z table or S function pnorm. However, "there are no such simple formulae for the T test. This is because the probability distribution of T when $\mu \neq \mu_o$ is very complicated" (9:263). Fortunately, the probabilities of T can be computed by the ISTP using IMSL. Consequently, the ISTP provides the user with the opportunity of computing the power for the T test and obtaining the power function. Macro ?power which can be called by menu selection or by direct macro selection, is listed in Table 18.

Extension to FORTRAN. A double integration routine DBLIN (11:DBLIN 1-3) is used to estimate a bivariate CDF from its known bivariate PDF. The non-central T distribution probability routine MDTN (12:MDTN 1-2) is executed to compute the CDF and estimate the PDF of a non-central T random variable. These FORTRAN routines are linked to S through an S interface routine (4:3-40). For example, by letting bvn.i be the interface routine and bvn.f be the FORTRAN routine and the new function be called bvnorm in the interface routine, then the new function bvnorm can be defined by the following procedure (page 130):

TABLE 18

DESCRIPTIONS OF MACROS FOR THE POWER FUNCTIONS

Type of Function	Direction Macro (arguments)	Arguments
Computing Power (?pwrone)	Right tail test ?pwr1(m0,m1,s,n,a)	M0 : μ_o
	Left tail test ?pwr2(m0,m1,s,n,a)	M1 : μ_1
	Two tail test ?pwr3(m0,m1,s,n,a)	s : Sample St-Dev.
Display Power Function (?pwrtwo)	Right tail test ?pwra(m0,s,n,mn,mx)	n : Sample size
	Left tail test ?pwrb(m0,s,n,mn,mx)	mn : Min μ for plot
	Two tail test ?pwrc(m0,s,n,mn,mx	mx : Max μ for plot

```
% S CHAPTER
S chapter initialized in current directory

% S FUNCTION bvnorm bvn.i bvn.f /usr/lib/libimsls.a
bvnorm initialized.

% S MAKE bvnorm
i.bvnorm.C
bvn.f:
   zbvnor:
bvn.f:
   bvnormf:
   fl:
   ay:
   by:
bvnorm loaded
%
```

Just before using the new function bvnorm in S, the chapter command must be issued.

V. Conclusions and Recommendations

This chapter describes the lessons learned during the development of the ISTP and makes recommendations for current uses and future research involving the package.

Conclusions

The ISTP was developed to fulfill the immediate need for a tutorial type package to support AFIT statistics courses. Furthermore, the ISTP was designed to accommodate users' demands for different levels of involvement with the package. The student who wants to use the ISTP package in a passive learning mode can do this by taking advantage of its menu capabilities. On the other hand, the individual who wants access to the ISTP to expand his knowledge of statistics can study such concepts graphically using the macros of the ISTP as well as other functions in S. Finally, if the user has a very strong quantitative background and computer experience, he is encouraged to modify and extend the core modules of the ISTP package. For purpose of discussion, the aforementioned types of user involvement with S and the ISTP will be classified as level 1, level 2, and level 3, respectively.

During level 1 involvement with the ISTP, the student can play with the ISTP to discover what shape a

member of a probability distribution family has or to reveal the significance of a transformation of a random variable on the testing process. Facilitating participation at this level is critical, because the traditional lecture/ lab approach to learning statistics fails to motivate the student's desire to learn statistics the way this tutorial package can. The ISTP allows the user to approach statistical thinking in a way that minimizes the fear of failure by continuous feedback facilitating immediate rectification of a mistaken concept.

Once motivated by level 1 experiences, the user can involve himself in an active learning process by directly invoking selected macros of the ISTP. This direct use of macros offers the user an opportunity to master and embellish the tutorial material encompassed by the ISTP package. Passive involvement with the ISTP, promoted by level 1 learning experiences, can become exceedingly burdensome to the sophisticated user. Direct calling of macros, in lieu of using menus, builds the user's confidence in his mastery of statistics and promotes a creative relationship with the package. The breadth of coverage of available ISTP macros encourages users, at level 2, to develop and analyze the simulated data based on real-world scenarios.

Level 3 involvement allows users to create their own library of macros. To do this, the user must have the capability of programming in either FORTRAN or C and have

acquired an exceptional education in mathematical statistics. It is at this level of involvement that genuine pedagogical research can be conducted. Future level 3 research should continue to create a pedagogically attractive learning environment for level 1 and level 2 users.

Recommendations

The final two sections of this thesis will present (1) how the user can take better advantage of the ISTP's current capability, and (2) additional areas future research should address.

Recommendations for the User.

1. It is recommended that macros for the four subject areas covered by the ISTP be used in conjunction with one another so that output from one topic area can be examined in light of the results produced by macros in other areas. For example, after generating random variates from a certain probability distribution, the user could compare the histogram of the random variates with the theoretical shape of the probability distribution to determine the adequacy of the fit.

2. It is recommended that the ISTP be routinely accessible to any user. Adding the macros of the ISTP to the S system directory will greatly expand the current capabilities of S.

3. It is recommended that users evaluate the reasonableness of inputs they give to the ISTP. For example, if a student tries to transform random variables by requesting an extremely narrow interval width or very large numbers of intervals, the time it takes to solve a problem may be significant. The user should keep in mind that the purpose of the ISTP is to provide conceptual understanding for each topic, and not to provide extended computational capability.

4. Users should be encouraged to develop their own macros as they gain experience with S and the ISTP. Such macros can be stored in the position 2 library.

5. It is recommended that all users try to reach at least level 2 involvement with the ISTP. S is a very good tool for doing statistics. When users can directly access macros of the ISTP, S's advanced features can play a more significant role in the study of statistical concepts. Therefore, once a user gets comfortable using the ISTP as a menu-driven system, it is recommended that he make direct use of macros.

Recommendations for Future Research. The capability of the ISTP should be extended in all areas addressed by the current version. Additional macro packages should be built to cover the area of axiomatic probability, Bayesian estimation and inference, and the full spectrum of multivariate technologies and probability distributions.

The ISTP should be equipped to provide graphical output to all sorts of graphics devices, from standard CRT terminals to the more sophisticated work stations currently available in AFIT. Color graphics, animation and windowing need to be introduced to provide a foundation for extraordinary enhancement of level 1 and level 2 involvement with the ISTP.

Finally, to ensure that the ISTP runs fast enough to cope with the computations needs of forthcoming graphic packages, consideration should be given to porting the entire ISTP package to super computers available in AFIT and the ASD computer center at Wright Patterson Air Force Base.

Appendix A: Macros for Probability Distributions

Table of Contents

```
MACRO mber(
x/?PROMPT(X value; 0 or 1 : )/,
p/?PROMPT(P of X = 1      : )/)
#
# This is a macro to compute the PMF for Bernoulli Distribution.
#
({
    ?T(x)_x
    ?T(p)_p
    ?T_(?T(p)^?T(x))*((1-?T(p))^(1-?T(x)))
    rm(?T(x),?T(p),?T,value=?T)
 })
END
```

```
MACRO rber(
n/?PROMPT(How many R.V. ?       :)/,
p/?PROMPT(Probability of X = 1 :)/)
({
#
# This is a Macro to generate random variate for
# Bernoulli distribution.
#
  ?T(n)_n
  ?T(p)_p
  ?T(rn)_runif(?T(n))
  ?T_rep(0,?T(n))
  ?T[?T(rn)<=?T(p)]_1
  ?T[?T(rn)>?T(p)]_0
  rm(?T(n),?T(p),?T(rn),?T,value=?T)
})
END
```

```
MACRO mgeo(
x/?PROMPT(Number of Failure before 1st Success : )/,
p/?PROMPT(Probability of Success                : )/)
#
# This is Macro for Geometric Distribution
# 0 <= p <= 1
# x = 0,1,2,3,....,infinite integer
# where, x is the number of failure before getting first
# success.
#
({
   ?T(x)_x
   ?T(p)_p
   ?T_?T(p)*((1-?T(p))^?T(x))
   rm(?T(x),?T(p),?T,value=?T)
})
END
```

```
MACRO geo(
x1/?PROMPT(Min Number of Failure before 1st Success : )/,
x2/?PROMPT(Max Number of Failure before 1st Success : )/,
p/?PROMPT(Probability of Success                    : )/)
#
# This Macro is to compute CDF for
# geometric distribution.
#
({
   ?T_sum(?mgeo((x1:x2),p))
   rm(?T, value=?T)
})
END
```

```
MACRO rgeo(
n/?PROMPT(How many R.V. ?          : )/,
p/?PROMPT(Probability of Success   : )/)
({
#
# This is a Macro to generate random variate for
# Geometric distribution.
#
  ?T(n)_n
  ?T(p)_p
  ?T(x)_max(30,ceiling(5/?T(p)))
#
# To compute pmf, reasonable range should be determined.
# The longer range, the better precision. However, there
# should be trade off between precision and efficiency.
# From 0 to 5/P will give sufficient  precision,
# because the cummulative probability up to 5/P is
# approximately 1 except very small P. Such a case of
# small lambda, pmf will be computed from 0 to 30,
# instead 5/P.
#
  ?T(pm)_?mgeo(0:?T(x),?T(p))
  ?T(cm)_cumsum(?T(pm))
  ?T(cm)_?T(cm)[?T(cm)<=1]
#
# PMF for Geometric probability can be computed from the
# Macro ?mgeo, and then compute its cummulative probability
# by using "cumsum"
#
  ?T(rn)_runif(?T(n))
#
# Generate N random numbers to compare these numbers to the cdf,
# then find the position in the vector and subtract by 1 to make
# them the geometric variable, which are x = 0,1,2,...n,...
#
  ?T_?cmpsn(?T(cm),?T(rn))-1
  rm(?T(n),?T(p),?T(pm),?T(cm),?T(rn),?T,value=?T)
})
END
```

```
MACRO ddunif(
x/?PROMPT(X value       : )/,
l/?PROMPT(Lower bound   : )/,
u/?PROMPT(Upper bound   : )/)
({
#
# This is a Macro to compute the pmf for discrete
# Uniform distribution.
#
  ?T(x)_x
  ?T(l)_l
  ?T(u)_u
  ?T_ifelse( (?T(x)>=?T(l) & ?T(x)<=?T(u)), 1/(?T(u)-?T(l)+1), 0 )
  rm (?T(x),?T(l),?T(u),?T,value=?T)
})
END
```

```
MACRO pdunif(
x/?PROMPT(Quantile value : )/,
l/?PROMPT(Lower bound    : )/,
u/?PROMPT(Upper bound    : )/)
({
#
# This is a Macro to compute the cdf for discrete
# Uniform distribution.
#
  ?T(x)_x
  ?T(l)_l
  ?T(u)_u
  ?T_ifelse( ?T(x)<?T(l), 0, ifelse(?T(x)<?T(u),
             (?T(x)-?T(l)+1)/(?T(u)-?T(l)+1), 1) )
  rm (?T(x),?T(l),?T(u),?T,value=?T)
})
END
```

```
MACRO rdunif(
n/?PROMPT(How many R.V. ?     : )/,
l/?PROMPT(Lower bound         : )/,
u/?PROMPT(Upper bound         : )/)
#
# This Maco is to generate random variate for
# discrete uniform distribution.
#
({
   rv_runif(n)
   ?T(l)_l
   ?T(u)_u
   int_?T(u)-?T(l)+1
   mass_1/int
   pmf_rep(mass, int)
   cdf_cumsum(pmf)
   ?T_?cmpsn(cdf, rv)+?T(l)-1
   rm(?T(l),?T(u),rv,int,mass,pmf,cdf,?T,value=?T)
})
END
```

```
MACRO mbin(
x/?PROMPT(Number of Success : )/,
n/?PROMPT(Number of Trial   : )/,
p/?PROMPT(P of Success      : )/)
#
# This is a macro to compute the PMF for binomial probability
#
({
  ?T(x)_x
  ?T(n)_n
  ?T(p)_p
  ?T_ifelse( ?T(x)<?T(n)+1, exp((lgamma(?T(n)+1)-lgamma(?T(x)+1)-
     lgamma(?T(n)-?T(x)+1))+(?T(x)*log(?T(p)))+
     ((?T(n)-?T(x))*log(1-?T(p)))) ,0)
# When x greater than n, 0 will be the answer.
    rm(?T(x),?T(n),?T(p),?T,value=?T)
 })
END
```

```
MACRO bin(
b/?PROMPT(Smallest Number of S : )/,
e/?PROMPT(Largest Number of S  : )/,
n/?PROMPT(Number of Trial      : )/,
p/?PROMPT(Probability of S     : )/)
#
# This is a Macro to compute CDF for binomial probability
#
({
  ?T_sum(?mbin(b:e,n,p))
  rm(?T,value=?T)
 })
END
```

```
MACRO rbin(
m/?PROMPT(How many R.V. ?         :)/,
n/?PROMPT(Number of Trial         :)/,
p/?PROMPT(Probability of Success :)/)
({
#
# this is a Macro to generate random variate for
# Binomial distribution.
#
  ?T(m)_m
  ?T(n)_n
  ?T(p)_p
#
# 1 trial of binomial probability is same as the Bernouli probability,
# therefore, n trial of binomial is same as the sum of Bernouli r.v.
#
  ?T(rn)_?rber(?T(n)*?T(m),?T(p))
  ?T(rn)_matrix(?T(rn),?T(m),?T(n),byrow=TRUE)
  ?T_?row(?T(rn),sum)
  rm(?T(m),?T(n),?T(p),?T(rn),?T,value=?T)
})
END
```

```
MACRO mnegbin(
x/?PROMPT(Number of Failure   : )/,
r/?PROMPT(Number of Success   : )/,
p/?PROMPT(Probability of S    : )/)
#
# This is a macro to compute the PMF for negative binonial
# probability distribution
#
({
 ?T(x)_x
 ?T(r)_r
 ?T(p)_p
 ?T_exp(lgamma(?T(x)+?T(r))-lgamma(?T(x)+1)-lgamma(?T(r))+
    ?T(r)*log(?T(p))+?T(x)*log(1-?T(p)))
  rm(?T(x),?T(p),?T(r),?T,value=?T)
 })
END
```

```
MACRO negbin(
b/?PROMPT(Smallest Number of Failure : )/,
e/?PROMPT(Largest  Number of Failure : )/,
r/?PROMPT(Number of Success          : )/,
p/?PROMPT(Probability of Success     : )/)
#
#
# This is a macro to compute the CDF for negative binonial
# probability distribution
#
({
  ?T_sum(?mnegbin(b:e,r,p))
  rm(?T,value=?T)
 })
END
```

```
MACRO rnegbin(
n/?PROMPT(How many R.V. ?         : )/,
r/?PROMPT(Number of Success       : )/,
p/?PROMPT(Probability of Success : )/)
({
#
# This is a Macro to generate random variate for
# Negative Binomial distribution.
#
  ?T(n)_n
  ?T(r)_r
  ?T(p)_p
  ?T(m)_max(30,ceiling(?T(r)/?T(p))*5)
  ?T(pm)_?mbin(?T(r),(?T(r):(?T(r)+?T(m))),?T(p))
#
# PMF for negative binomial probability can be computed
# from binomial probability, let's say, we want to know
# the probability that R success will accur out of R+m trial
# which each trial has P probability of success, then the m
# is the number of failure prior to the R th success.
# Therefore, each pmf for from m=0 to m=infinitive can be computed.
#
  ?T(cm)_cumsum(?T(pm))
#
# The sum of pmf for the binomial probability is 1/P when we compute
# the pmf from m=0 to m=inf. However, if we compute the pmf from
# m=0 to m=(5*R/P), the sum of pmf can be approximately 1/P.
# In case of small m, 30 will be replaced.
#
  ?T(cm)_?T(cm)*?T(p)
#
# By multiplying P, the max cdf = 1.
#
  ?T(cm)_?T(cm)[?T(cm)<=1.0]
  ?T(rn)_runif(?T(n))
#
# Generate N random numbers to compare these numbers to the cdf,
# so that find the position which will ensure P[i-1] < X <= P[i].
#
  ?T_?cmpsn(?T(cm),?T(rn))-1
#
# After finding out the position in the vector which will ensure
# P[i-1] < X <= P[i], we subtract by 1 to make them the Negative
# Binomial Variable.
#
  rm(?T(p),?T(r),?T(n),?T(pm),?T(rn),?T(m),?T(cm),?T,value=?T)
})
END
```

```
MACRO mhyper(
x/?PROMPT(Number of Success in Sample  : )/,
n/?PROMPT(Sample size                  : )/,
M/?PROMPT(Success in Population        : )/,
N/?PROMPT(Population Size              : )/)
#
# This is a macro to call another macro to compute the PMF for
# hypergeometric probability distribution.
#
({
  ?T_?mhyper1(x,n,M,N)
  rm(?T,value=?T)
})
END
```

```
MACRO mhyper1(x,n,M,N)
#
#
# This is a macro to compute the PMF for hypergeometric
# probability distribution
#
({
?T_ ifelse(((x<=min(n,M)) & (x>=(M+n-N))),
      ?T_exp(lgamma(M+1)+lgamma(N-M+1)+lgamma(N-n+1)+lgamma(n+1)
      -lgamma(M-x+1)-lgamma(x+1)-lgamma(N-M-n+x+1)-lgamma(n-x+1)
      -lgamma(N+1)),0)
#
# Requirement for X: (M+n-N) <= x <= min(n,M)
# otherwise, the probability is always zero
#
   rm(?T,value=?T)
 })
END
```

```
MACRO hyper(
x1/?PROMPT(Min Success in Sample  : )/,
x2/?PROMPT(Max Success in Sample  : )/,
n/?PROMPT(Sample size             : )/,
M/?PROMPT(Success in Population   : )/,
N/?PROMPT(Population Size         : )/)
#
# This is a macro to compute the CDF for hypergeometric
# probability distribution
#
({
  ?T_sum(?mhyper1(x1:x2,n,M,N))
  rm(?T,value=?T)
 })
END
```

```
MACRO rhyper(
m/?PROMPT(How many R.V. ?          : )/,
n/?PROMPT(Sample Size              : )/,
M/?PROMPT(Success in Population    : )/,
N/?PROMPT(Population Size          : )/)
({
#
# This is a Macro to generate random variate for
# Hypergeometric distribution.
#
  ?T(m)_m
  ?T(n)_n
  ?T(M)_M
  ?T(N)_N
  ?T(mn)_max(0,?T(n)-?T(N)+?T(M))
  ?T(mx)_min(?T(n),?T(M))
  ?T(pm)_?mhyper1(?T(mn):?T(mx),?T(n),?T(M),?T(N))
#
# PMF from possible minimum number of success to maximum
# success is computed.
#
  ?T(cm)_cumsum(?T(pm))
#
# Each cumulative probability is computed by cumsum function.
#
  ?T(rn)_runif(?T(m))
  ?T_?cmpsn(?T(cm),?T(rn))+?T(mn)-1
#
# Each random number will be compared to the CDF and returned
# in terms of the position in the CDF veltor, therefore, if we
# add the possible minimum number of success and subtract by 1,
# we can get the random variate for Hypergeometric distribution.
#
  rm(?T(m),?T(N),?T(M),?T(n),?T(pm),?T(rn),?T(mn),?T(mx),
     ?T(cm),?T,value=?T)
})
END
```

```
MACRO mpoiss(
x/?PROMPT(X value    : )/,
r/?PROMPT(Lambda     : )/)
#
# This is a macro to compute the PMF for poisson
# probability distribution
#
({
   ?T(x)_x
   ?T(r)_r
   ?T_exp(-?T(r)+?T(x)*log(?T(r))-lgamma(?T(x)+1))
  rm(?T(x),?T(r),?T,value=?T)
 })
END
```

```
MACRO poiss(
x1/?PROMPT(Min X value    : )/,
x2/?PROMPT(Max X value    : )/,
r/?PROMPT(Lambda          : )/)
#
# This is a macro to compute the CDF for poisson
# probability distribution
#
({
  ?T_sum(?mpoiss(x1:x2,r))
  rm(?T,value=?T)
 })
END
```

```
MACRO rpoiss(
n/?PROMPT(How many R.V. ?  : )/,
lm/?PROMPT(Lambda          : )/)
({
#
# This is a Macro to generate random variate for
# Poisson distribution.
#
  ?T(n)_n
  ?T(l)_lm
  ?T(x)_max(30,ceiling(5*?T(l)))
#
# To compute pmf, reasonable range should be determined.
# The longer range, the better precision. However, there
# should be trade off between precision and efficiency.
# From 0 to 5*lambda will give sufficient  precision,
# because the cummulative probability up to 5*lambda
# is approximately 1 except very small lambda. Such a
# case of small lambda, pmf will be computed from 0 to 30,
# instead 5*lambda.
#
  ?T(pm)_?mpoiss(0:?T(x),?T(l))
  ?T(cm)_cumsum(?T(pm))
  ?T(cm)_?T(cm)[?T(cm)<=1]
#
# PMF for Poisson probability can be computed from the
# Macro ?mpoiss, and then compute its cummulative probability
# by using function cumsum.
#
  ?T(rn)_runif(?T(n))
#
# Generate N random numbers to compare these numbers to the cdf,
# then find the position in the vector so as to determin the
# random variate by subtracting 1 from the position.
#
  ?T_?cmpsn(?T(cm),?T(rn))-1
  rm(?T(n),?T(l),?T(pm),?T(cm),?T(rn),?T,value=?T)
})
END
```

```
MACRO mreylam(
x/?PROMPT(X value 0, 1 or 2       : )/,
p1/?PROMPT(Probability of X = 0   : )/,
p2/?PROMPT(Probability of X = 1   : )/)
({
#
# This is a Macro to compute pmf for ReyLamb distribution,
# which has two parameters and possible outcomes are 0, 1, and 2.
# P1 is a probability associated with 0, P2 is a probability
# associated with 1 and (1-P1-P2) is a probability associated
# with outcome 3. P1+P2 should be equal to or less than 1.
#
  ?T(x)_x
  ?T(1)_p1
  ?T(2)_p2
  ?T_(1-?T(x))*?T(1)+?T(x)*?T(2)+?T(x)*(1-?T(x))*(1.5*?T(2)-.5)
  rm_(?T(x),?T(1),?T(2),?T,value=?T)
})
END



MACRO rreylam(
n/?PROMPT(How many R.V. ?         : )/,
p1/?PROMPT(Probability of X = 0   : )/,
p2/?PROMPT(Probability of X = 1   : )/)
({
#
# This is a Macro to generate random variate for
# ReyLamb distribution.
#
  ?T(n)_n
  ?T(p1)_p1
  ?T(p2)_p2
  ?T(pm)_?mreylam(0:2,?T(p1),?T(p2))
  ?T(cm)_cumsum(?T(pm))
#
# PMF for ReyLamb probability can be computed from the
# Macro ?mreylam, and then compute its cummulative probability
# by using function cumsum.
#
  ?T(rn)_runif(?T(n))
#
# Generate N random numbers to compare these numbers to the cdf,
# then find the position in the vector so as to determin the
# random variate by subtracting 1 from the position.
#
  ?T_?cmpsn(?T(cm),?T(rn))-1
  rm_(?T(n),?T(p1),?T(p2),?T(pm),?T(cm),?T(rn),?T,value=?T)
})
END
```

```
MACRO dex(
x/?PROMPT(X value : )/,
lm/?PROMPT(Lambda  : )/)
#
# This is a macro to compute the PDF for exponential
# probability distribution
({
   ?T_?dex1(x,lm)
   rm_(?T, value=?T)
})
END
```

```
MACRO dex1(x,lm)
#
({
   ?T_lm*exp(-lm*x)
   rm_(?T,value=?T)
})
END
```

```
MACRO pex(
x/?PROMPT(Quantile : )/,
r/?PROMPT(Lambda   : )/)
#
# This is a macro to compute the CDF for exponential
# probability distribution
({
  ?T_?pex1(x,r)
  rm_(?T, value=?T)
})
END
```

```
MACRO pex1(x,r)
({
  ?T_1-(exp(-(r*x)))
  rm_(?T,value=?T)
})
END
```

```
MACRO qex(
p/?PROMPT(Cumulative Prob. : )/,
r/?PROMPT(Lambda           : )/)
#
# This is a macro to compute the Qunatile for exponential
# probability distribution
({
  ?T_(-log(1-p))/r
  rm(?T,value=?T)
})
END
```

```
MACRO rex(
n/?PROMPT(How many R.V. ?  : )/,
r/?PROMPT(Lambda           : )/)
({
#
# This is a Macro to generate random variates for
# exponential distribution.
#
  ?T_?qex(runif(n),r)
  rm(?T,value=?T)
})
END
```

```
MACRO gamfun(
x/?PROMPT(X value  : )/,
a/?PROMPT(Alpha    : )/,
b/?PROMPT(Beta     : )/)
#
# This is a macro to call another macro to compute the pdf for
# gamma probability distribution.
#
({
  ?T_?gamfun1(x,a,b)
  rm(?T,value=?T)
})
END
```

```
MACRO gamfun1(x,a,b)
#
# This is a macro to compute the PDF for gamma
# probability distribution with two parameters
({
  ?T_ifelse(x!=0,exp((a-1)*log(x)-x/b-a*log(b)-lgamma(a)),
     ifelse(a>1,0, ifelse(a==1,1/b,dgamma(0,.1))))
# If x=0, when a>1 pdf = 0, when a=1 pdf=1/b, when a<1 pdf=inf
  rm(?T,value=?T)
})
END
```

```
MACRO dweib(
x/?PROMPT(X value          : )/,
a/?PROMPT(Shape Parameter : )/,
b/?PROMPT(Scale Parameter : )/)
#
# This is a macro to call another macro to compute the PDF for weibull
# probability distribution.
#
({
  ?T_?dweib1(x,a,b)
  rm(?T,value=?T)
})
END
```

```
MACRO dweib1(x,a,b)
#
# This is a macro to compute the PDF for weibull
# probability distribution
({
  ?T_ifelse(x!=0,exp(log(a)+(a-1)*log(x)-(x/b)^a-a*log(b)),
     ifelse(a>1,0, ifelse(a==1,1/b,dgamma(0,.1))))
# If x=0, when a>1 pdf = 0, when a=1 pdf=1/b, when a<1 pdf=inf
  rm(?T,value=?T)
})
END
```

```
MACRO pweib(
x/?PROMPT(Quantile value     : )/,
a/?PROMPT(Shape Parameter    : )/,
b/?PROMPT(Scale Parameter    : )/)
#
# This is a macro to compute the CDF for weibull
# probability distribution
({
  ?T(x)_x
  ?T(a)_a
  ?T(b)_b
  ?T_1-(exp(-((?T(x)/?T(b))^?T(a))))
  rm(?T,value=?T)
})
END
```

```
MACRO qweib(
p/?PROMPT(Cumulative Prob.    : )/,
a/?PROMPT(Shape Parameter     : )/,
b/?PROMPT(Scale Parameter     : )/)
#
# This is a macro to compute the Quantile for weibull
# probability distribution
({
  ?T(p)_p
  ?T(a)_a
  ?T(b)_b
  ?T_exp(log(?T(b))+(1/?T(a))*(log(-log(1-?T(p)))))
  rm(?T,value=?T)
})
END
```

```
MACRO rweib(
n/?PROMPT(How many R.V. ?     : )/,
a/?PROMPT(Shape Parameter     : )/,
b/?PROMPT(Scale Parameter     : )/)
({
#
# This Macro is to generate Weibull Random Variate.
#
  ?T(n)_n
  ?T(a)_a
  ?T(b)_b
  ?T_?qweib(runif(?T(n)),?T(a),?T(b))
  rm(?T(n),?T(a),?T(b),?T,value=?T)
})
END
```

```
MACRO dtrian(
xv/?PROMPT(X vaulue    : )/,
av/?PROMPT(Lower Bound : )/,
cv/?PROMPT(Upper Bound : )/,
bv/?PROMPT(Mode        : )/)
({
#
# This Macro is to compute pdf for Triangular distribution.
# where, x : quantile value ( a <= x <= c )
#        av: lower bound
#        cv: upper bound
#        bv: mode
#
  ?T(x)_xv
  ?T(a)_av
  ?T(u)_cv
  ?T(b)_bv
  if(?T(x)<?T(a) | ?T(x)>?T(u)) fatal("X value can't be smaller than
             lower bound or greater than upper bound")
  if(?T(a)>?T(b) | ?T(u)<?T(b)) fatal("Mode can't be smaller than
             lower bound or greater than upper bound")
  ?T_ifelse( (?T(b)==?T(u)) | ( ?T(x)<=?T(b) & ?T(a)!=?T(b) ),
       2*(?T(x)-?T(a))/((?T(b)-?T(a))*(?T(u)-?T(a))),
       2*(?T(u)-?T(x))/((?T(u)-?T(b))*(?T(u)-?T(a))))
  rm (?T(x),?T(a),?T(u),?T(b),?T,value=?T)
})
END
```

```
MACRO ptrian(
xv/?PROMPT(Quantile    : )/,
av/?PROMPT(Lower Bound : )/,
cv/?PROMPT(Upper Bound : )/,
bv/?PROMPT(Mode        : )/)
({
#
# This Macro is to compute cdf for Triangular distribution.
# where, x : quantile value ( a <= x <= c )
#        av: lower bound
#        cv: upper bound
#        bv: mode
#
  ?T(x)_xv
  ?T(a)_av
  ?T(u)_cv
  ?T(b)_bv
  if(?T(x)<?T(a) | ?T(x)>?T(u)) fatal("X value can't be smaller than
           lower bound or greater than upper bound")
  if(?T(a)>?T(b) | ?T(u)<?T(b)) fatal("Mode can't be smaller than
           lower bound or greater than upper bound")
  ?T_ifelse(?T(x)<=?T(b), (?T(x)-?T(a))^2/((?T(b)-?T(a))*(?T(u)-?T(a))),
     1-(?T(u)-?T(x))^2/((?T(u)-?T(b))*(?T(u)-?T(a))))
  rm (?T(x),?T(a),?T(b),?T(u),?T,value=?T)
})
END
```

```
MACRO qtrian(
pv/?PROMPT(Cum. Prob.  : )/,
av/?PROMPT(Lower Bound : )/,
cv/?PROMPT(Upper Bound : )/,
bv/?PROMPT(Mode        : )/)
({
#
# This Macro is to compute quantile for Triangular distribution.
# where, pv: cummulative probability (0 <= p <=1)
#        av: lower bound
#        cv: upper bound
#        bv: mode
#
  ?T(p)_pv
  ?T(a)_av
  ?T(u)_cv
  ?T(b)_bv
  if(?T(p)<0 | ?T(p)>1) fatal("P value can't be smaller than 0)
             or greater 1")
  if(?T(a)>?T(b) | ?T(u)<?T(b)) fatal("Mode can't be smaller than
             lower bound or greater than upper bound")
  ?T_ifelse(?T(p)<=(?T(b)-?T(a))/(?T(u)-?T(a)), ?T(a)+sqrt(?T(p)*
           ((?T(b)-?T(a))*?T(u)-?T(a)*?T(b)+?T(a)^2)),
           ?T(u)-sqrt((1-?T(p))*(?T(u)-?T(b))*(?T(u)-?T(a))))
  rm (?T(p),?T(a),?T(u),?T(b),?T,value=?T)
})
END
```

```
MACRO rtrian(
nv/?PROMPT(How many R.V. ?  : )/,
av/?PROMPT(Lower Bound      : )/,
cv/?PROMPT(Upper Bound      : )/,
bv/?PROMPT(Mode             : )/)
({
#
# This Macro is to generate random variates for Triangular
# distribution,
# where, nv: number of random variates
#        av: lower bound
#        cv: upper bound
#        bv: mode
#
  ?T(n)_nv
  ?T(a)_av
  ?T(u)_cv
  ?T(b)_bv
  if(?T(a)>?T(b) | ?T(u)<?T(b)) fatal("Mode can't be smaller than
             lower bound or greater than upper bound")
  ?T_?qtrian(c(runif(?T(n))),?T(a),?T(u),?T(b))
  rm(?T(n),?T(a),?T(u),?T(b),?T,value=?T)
})
END
```

```
MACRO dbinorm(
mx/?PROMPT(MU 1        : )/,
sx/?PROMPT(Sigma 1     : )/,
my/?PROMPT(MU 2        : )/,
sy/?PROMPT(Sigma 2     : )/,
r/?PROMPT(Coefficient of Correlation : )/,
x/?PROMPT(Quantile X : )/,
y/?PROMPT(Quantile Y : )/)
({
   ?dbinorm1(mx,sx,my,sy,r,x,y)
})
END
```

```
MACRO dbinorm1(mx,sx,my,sy,r,x,y)
#
# This is a macro to get the PDF for both Independent and
# Dependent Bivariate Normal Probability
#
({
  k1_1/(2*pi*sx*sy*sqrt(1-r^2))
  k2_-1/(2*(1-r^2))
  ?T_k1*exp(k2*(((x-mx)/sx)^2-2*r*((x-mx)/sx)*((y-my)/sy)
    _ +((y-my)/sy)^2))
  rm(?T,value=?T)
})
END
```

```
MACRO rbinorm(
nv/?PROMPT(How many R.V. ?   : )/,
mxv/?PROMPT(Mean of X         : )/,
sxv/?PROMPT(Sigma of X        : )/,
myv/?PROMPT(Mean of Y         : )/,
syv/?PROMPT(Sigma of Y        : )/,
rho/?PROMPT(Correlation       : )/)
({
#
# This is to generate random variate for Bivariate
# Normal distribution.
#
   ?T(n)_nv
   ?T(mx)_mxv
   ?T(sx)_sxv
   ?T(my)_myv
   ?T(sy)_syv
   ?T(rh)_rho
   ?T(rh)_c(1,?T(rh),?T(rh),1)
   ?T(rh)_matrix(?T(rh),2,2,byrow=TRUE)
   ?T(m)_c(?T(mx),?T(my))
   ?T(sd)_c(?T(sx),?T(sy))
   ?T(v)_?T(rh)*outer(?T(sd),?T(sd))
   ?T(x)_matrix(rnorm(?T(n)*2),?T(n),2)
   ?T_?T(x)%*chol(?T(v))+matrix(?T(m),?T(n),2)
   ?T_t(?T)
   rm(?T(n),?T(mx),?T(sx),?T(my),?T(sy),?T(rh),
      ?T(m),?T(sd),?T(v),?T(x),?T,value=?T)
})
END
```

```
MACRO rbigamma(
n/?PROMPT(How many R.V. ? : )/,
ax/?PROMPT(Alpha 1 : )/,
ay/?PROMPT(Alpha 2 : )/)
#
# This is a Macro to generate Random Deviates for
# Independent Standard Gamma Probability
#
({
?T(n)_n
?T(ax)_ax
?T(ay)_ay
?T(q)_runif(?T(n))
#
# ?T(q) : Random CDF for bivariate gamma probability
#
?T(cx)_runif(?T(n),?T(q),1)
#
# ?T(cx) : Random CDF for X
#
?T(cy)_?T(q)/(?T(cx))
#
# ?T(cy) : Random CDF for Y
#
?T(qx)_qgamma(?T(cx),?T(ax))
?T(qy)_qgamma(?T(cy),?T(ay))
?T_rbind(?T(qx),?T(qy))
rm(?T(ax),?T(ay),?T(n),?T(q),?T(cx),
  ?T(cy),?T(qx),?T(qy),?T,value=?T)
})
END
```

```
MACRO cmpsn(x,a)
({
#
# This is Macro to find out the first position in the vector
# which will be greater than the given "a" value.
# If given value is greater than any value in x, len(a)+1 be
# the return value.
# Therefore, it is usefull to find the quantile value
# for discrete distribution.
#
  ?T(x)_x
  ?T(a)_a
  ?T(x)_matrix(?T(x),len(?T(a)),len(?T(x)),byrow=T)
  ?T(x)_ifelse(?T(x)>=?T(a), 0 , 1)
  ?T_?row(?T(x),sum)+1
  rm(?T(x),?T(a),?T,value=?T)
})
END
```

```
MACRO dist
#
# This MAcro is to select univariate or bivariate
# distribution.
#
({
    item_c("Univariate distribution",
           "Bivariate  distribution")
    action_c("?uvd",
             "?bvd")
    menu( item, action)
    rm( item, action)
})
END
```

```
MACRO uvd
({
   item_c("DISCRETE   DISTRIBUTION",
          "CONTINUOUS DISTRIBUTION",
          "QUIT")
   action_c("?dis","?cont","q")
   menu( item , action)
   rm( item, action)
})
END
```

```
MACRO dis
({
  item_c("BINOMIAL          DISTRIBUTION",
         "NEGATIVE BINOMIAL DISTRIBUTION",
         "HYPERGEOMETRIC    DISTRIBUTION",
         "UNIFORM           DISTRIBUTION",
         "POISSON           DISTRIBUTION",
         "GO TO PREVIOUS STEP")
  action_c("?binom",
           "?negbinom",
           "?hypergeo",
           "?uniform",
           "?poisson",
           "?uvd")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO cont
({
  item_c("NORMAL      DISTRIBUTION",
         "UNIFORM     DISTRIBUTION",
         "LOG-NORMAL  DISTRIBUTION",
         "T           DISTRIBUTION",
         "EXPONENTIAL DISTRIBUTION",
         "GAMMA       DISTRIBUTION",
         "WEIBULL     DISTRIBUTION",
         "BETA        DISTRIBUTION",
         "CHISQUARE   DISTRIBUTION",
         "F           DISTRIBUTION",
         "GO TO PREVIOUS STEP")
  action_c("?normal",
           "?cuniform",
           "?lognorm",
           "?tdis",
           "?expon",
           "?gammaset",
           "?weibull",
           "?betaset",
           "?chisquare",
           "?fdis",
           "?uvd")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO binom
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?binplot",
         "?obinplot",
         "?dis")
menu( item,action)
rm( item, action)
})
END




MACRO binplot(
n/?PROMPT(N VALUE  : )/,
p/?PROMPT(P VALUE  : )/)
#
# This is a Macro to plot the binomial probability distribution
#
({
    par(mfrow=c(1,1), oma=c(0,2,0,0))
    ?T(n)_n
    ?T(p)_p
    ?T(x)_seq(0,?T(n),1)
    ?T(y)_?mbin(?T(x),?T(n),?T(p))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
    ?T(my)_max(0.05,max(?T(y)))
    plot(?T(x),?T(y),main="SHAPE OF BINOMIAL PROBABILITY DISTRIBUTION",
         sub=encode("N : ",?T(n),"  P : ",?T(p)), type="h",
         xlab="X-VALUE", ylab="",xlim=c(-1,?T(n)+1),ylim=c(0,?T(my)),
         err=-1)
    mtext(side=2, line=1, outer=T, "PMF")
    rm(?T(x),?T(y),?T(my),?T(p),?T(n))
 })
END
```

```
MACRO obinplot(n/?PROMPT(NUMBER OF PLOT(up to 4)    : )/,
nv/?PROMPT(N VALUE(ex)8,10,16       : )/,
pv/?PROMPT(P VALUE(ex)0.3,0.5,0.8  : )/)
#
# This is a Macro to compare the shape of binomial probability
# distribution with different parameters.
#
({
?T(n)_n
if(?T(n)<=2) par(mfrow=c(1,2), oma=c(0,0,2,0))
if(?T(n)> 2) par(mfrow=c(2,2), oma=c(0,0,2,0))
?T(nv)_c(nv)
?T(p)_c(pv)
?T(x)_seq(0,max(?T(nv)),1)
?T(y)_matrix(0,nrow=?T(n),ncol=len(?T(x)))
#
# Binomial probability exists only x < n+1
#
for ( i in 1:?T(n) )  {
  ?T(y)[i,(1:(?T(nv)[i]+1))]_?mbin(?T(x)[1:(?T(nv)[i]+1)],
  ?T(nv)[i],?T(p)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_max(0.05,max(?T(y)))
param_encode("N :", ?T(nv)," P :", ?T(p))
for ( i in 1: ?T(n) ) {
  plot(?T(x),?T(y)[i,],xlab="X VALUE",ylab="",type="h",
      main=param[i], ylim=c(0,?T(my)))
      }
mtext(side=3, line=0, outer=T,
"SHAPES OF BINOMIAL PROBABILITY DISTRIBUTIONS")
rm(?T(n),?T(x),?T(nv),?T(p),?T(y),?T(my))
})
END
```

```
MACRO negbinom
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?negbinplot",
         "?onegbinplot",
         "?dis")
menu(item,action)
rm(item, action)
})
END
```

```
MACRO negbinplot(
   r/?PROMPT(Number of Success       : )/,
   p/?PROMPT(Probability of Success  : )/,
   x/?PROMPT(Max X for Plot          : )/)
#
# This is a Macro to plot the shape of negative binomial probability
# distribution.
#
({
 par(mfrow=c(1,1), oma=c(2,2,0,0))
 ?T(r)_r
 ?T(p)_p
 ?T(x)_x
 ?T(x)_(0:?T(x))
 ?T(y)_?mnegbin(?T(x),?T(r),?T(p))
 ?T(my)_max(0.05,max(?T(y)))
 #
 # To prevent too small y-axis, set the y-axis limit at least 0.05
 #
 plot(?T(x), ?T(y), xlab="", ylab="", type="h",
      main="SHAPE OF NEGATIVE BINOMAIL DISTRIBUTION",
      ylim=c(0,?T(my)), sub="Number of Failure", err=-1)
 mtext(side=2, line=1, outer=T, "PMF")
 mtext(side=1, line=1, outer=T,
    encode("P : ",?T(p),"  NUMBER of SUCCESS : ",?T(r)))
 rm(?T(x),?T(y),?T(my),?T(p),?T(r))
 })
 END
```

```
MACRO onegbinplot(
n/?PROMPT(Number of Plot (Up to 4)           : )/,
rv/?PROMPT(Number of Success (ex) 5, 8, 12   : )/,
pv/?PROMPT(Probability of S  (ex) .2,.3,.5 : )/,
maxx/?PROMPT(Max X for Plot                 : )/)
#
# This is a Macro to compare the shape of negative binomial
# probability distribution with different parameters.
#
({
?T(n)_n
if(?T(n)<=2) par(mfrow=c(1,2), oma=c(0,0,2,0))
if(?T(n)> 2) par(mfrow=c(2,2), oma=c(0,0,2,0))
?T(rv)_c(rv)
?T(pv)_c(pv)
?T(x)_seq(0,maxx,1)
?T(y)_matrix(0,nrow=?T(n),ncol=len(?T(x)))
for ( i in 1:?T(n) )  {
  ?T(y)[i,]_?mnegbin(?T(x),?T(rv)[i],?T(pv)[i])
  }
?T(my)_max(0.05,max(?T(y)))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
param_encode("P :", ?T(pv)," # of S :", ?T(rv))
for ( i in 1: ?T(n) ) {
  plot(?T(x),?T(y)[i,],xlab="NUMBER of FAILURE",ylab="",type="h",
      main=param[i], ylim=c(0,?T(my)), err=-1)
      }
mtext(side=3, line=0, outer=T,
"SHAPES OF NEGATIVE BINOMIAL PROBABILITY DISTRIBUTIONS")
rm(?T(n),?T(x),?T(pv),?T(rv),?T(y),?T(my))
})
END
```

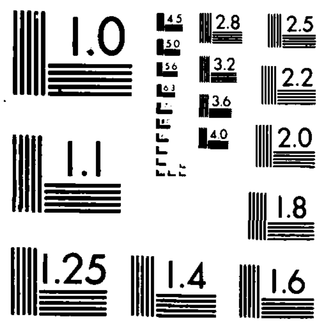
```
MACRO hypergeo
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?hyperplot",
         "?ohyperplot","?dis")
menu(item,action)
rm(item, action)
})
END
```

```
MACRO hyperplot(
      n/?PROMPT(Sample Size            : )/,
      M/?PROMPT(Success in Population : )/,
      N/?PROMPT(Population Size        : )/)
#
# This is a Macro to plot the hypergeometric
# probability distribution
#
({
   par(mfrow=c(1,1), oma=c(2,2,0,0))
   ?T(n)_n
   ?T(M)_M
   ?T(N)_N
   ?T(x)_(max(0,?T(M)+?T(n)-?T(N)):min(?T(M),?T(n)))
   ?T(y)_?mhyper1(?T(x),?T(n),?T(M),?T(N))
   ?T(my)_max(0.05,max(?T(y)))
 #
 # To prevent too small y-axis, set the y-axis limit at least 0.05
 #
   plot(?T(x), ?T(y), xlab="", ylab="", type="h",
        main="SHAPE of HYPERGEOMETRIC DISTRIBUTION",
        ylim=c(0,?T(my)), sub="Successes in Sample", err=-1)
   mtext(side=2, line=1, outer=T, "PMF")
   mtext(side=1, line=1, outer=T,
      encode("Pop' size:",?T(N)," Success in Pop':",?T(M),
      " Sample size:", ?T(n)))
   rm(?T(x),?T(y),?T(my),?T(N),?T(M),?T(n))
 })
END
```

```
MACRO ohyperplot(
n/?PROMPT(Number of Plot (Up to 4) : )/,
ss/?PROMPT(Sample Size              : )/,
ns/?PROMPT(Success in Population    : )/,
ps/?PROMPT(Population Size          : )/)
#
# This is a Macro to compare the shape of hypergeometric
# probability distribution with different parameters
#
({
?T(n)_n
if(?T(n)<=2) par(mfrow=c(1,2), oma=c(0,0,2,0))
if(?T(n)> 2) par(mfrow=c(2,2), oma=c(0,0,2,0))
?T(ss)_c(ss)
?T(ns)_c(ns)
?T(ps)_c(ps)
?T(x)_seq(0,min(max(?T(ns)),max(?T(ss))),1)
?T(y)_matrix(0,nrow=?T(n),ncol=len(?T(x)))
#
# Confirm the condition (M+n-N) <= x <= min(M,n)
#
for ( i in 1:?T(n) )  {
  ?T(y)[i,((max(1,(?T(ns)[i]+?T(ss)[i]-?T(ps)[i]+1))):
+ ((min(?T(ns)[i],?T(ss)[i]))+1))]_
+ ?mhyper1((max(0,(?T(ns)[i]+?T(ss)[i]-?T(ps)[i]))):
+  (min(?T(ns)[i],?T(ss)[i])),
+  ?T(ss)[i],?T(ns)[i],?T(ps)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_max(0.05,max(?T(y)))
param_encode("SS:",?T(ss)," NS:",?T(ns)," PS:",?T(ps))
for ( i in 1: ?T(n) ) {
  plot(?T(x),?T(y)[i,],xlab="NUMBER of SUCCESS in SAMPLE",
      ylab="",type="h",main=param[i],ylim=c(0,?T(my)),
      err=-1)
      }
mtext(side=3, line=0, outer=T,
"SHAPES OF HYPERGEOMETRIC PROBABILITY DISTRIBUTIONS")
rm(?T(n),?T(x),?T(ps),?T(ns),?T(ss),?T(y),?T(my))
})
END
```

AD-A174 297 DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL PACKAGE (ISTP) FOR LE..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF SYST.. K H CHUL 3/5

UNCLASSIFIED SEP 86 AFIT/GSM/ENC/86S-10 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
MACRO uniform
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?unifplot",
         "?ounifplot","?dis")
menu(item,action)
})
END
```

```
MACRO unifplot(
i/?PROMPT(LOWER VALUE  : )/,
e/?PROMPT(UPPER VALUE  : )/,
nx/?PROMPT(MIN for PLOT : )/,
xx/?PROMPT(MAX for PLOT : )/)
#
# This is a Macro to plot the discrete uniform
# probability distribution.
#
({
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(i)_i
  ?T(e)_e
  ?T(nx)_nx
  ?T(xx)_xx
  ?T(x)_(?T(nx):?T(xx))
  ?T(y)_?ddunif(?T(x),?T(i),?T(e))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
  ?T(my)_max(0.05,max(?T(y)))
  plot(?T(x),?T(y),xlab="X-VALUE",ylab="",type="h",
       main="SHAPE OF DISCRETE UNIFORM DISTRIBUTION",
       ylim=c(0,?T(my)), sub=encode("Lower Limit : ",
       ?T(i),"  Upper Limit : ",?T(e)), err=-1)
  mtext(side=2, line=1, outer=T, "PMF")
  rm(?T(x),?T(y),?T(my),?T(i),?T(e),?T(xx),?T(nx))
})
END
```

```
MACRO ounifplot(
n/?PROMPT(NUMBER OF PLOT(up to 4) : )/,
nv/?PROMPT(LOWER VALUE(ex)2,3,4    : )/,
pv/?PROMPT(UPPER VALUE(ex)5,8,9    : )/)
#
# This is a Macro to compare the shape of uniform probability
# distribution with different parameters
#
({
?T(n)_n
if(?T(n)<=2) par(mfrow=c(1,2), oma=c(0,0,2,0))
if(?T(n)> 2) par(mfrow=c(2,2), oma=c(0,0,2,0))
?T(nv)_c(nv)
?T(pv)_c(pv)
?T(x)_seq(min(?T(nv)),max(?T(pv)),1)
?T(y)_matrix(0,nrow=?T(n),ncol=len(?T(x)))
for ( i in 1:?T(n) )  {
  ?T(y)[i,]_?ddunif(?T(x),?T(nv)[i],?T(pv)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_max(0.05,max(?T(y)))
param_encode("LL :", ?T(nv)," UL :", ?T(pv))
for ( i in 1: ?T(n) ) {
  plot(?T(x),?T(y)[i,],xlab="X VALUE",ylab="",type="h",
      main=param[i],ylim=c(0,?T(my)))
      }
mtext(side=3, line=0, outer=T,
"SHAPES OF DISCRETE UNIFORM PROBABILITY DISTRIBUTIONS")
rm(?T(n),?T(x),?T(nv),?T(pv),?T(y),?T(my))
})
END
```

```
MACRO poisson
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?poisplot",
         "?opoisplot","?dis")
menu( item,action)
rm( item, action)
})
END
```

```
MACRO poisplot(
   l/?PROMPT(LAMBDA        : )/,
   x/?PROMPT(MAX X for PLOT : )/)
#
# This is a Macro to plot the poisson
# probability distribution
#
({
   par(mfrow=c(1,1), oma=c(0,2,0,0))
   ?T(l)_l
   ?T(x)_x
   ?T(x)_(0:?T(x))
   ?T(y)_?mpoiss(?T(x),?T(l))
   ?T(my)_max(0.05,max(?T(y)))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
   plot(?T(x),?T(y),xlab="X-VALUE",ylab="",ylim=c(0,?T(my)),
        main="SHAPE OF POISSON DISTRIBUTION",type="h",
        sub=encode("LAMBDA : ", ?T(l)))
   mtext(side=2, line=1, outer=T, "PMF")
   rm(?T(x),?T(y),?T(my),?T(l))
 })
END
```

```
MACRO opoisplot(
n/?PROMPT(NUMBER OF PLOT(up to 4) : )/,
nv/?PROMPT(LAMBDA (ex) 2,3,4     : )/,
pv/?PROMPT(MAX X for PLOT        : )/)
#
# This is a Macro to compare the shape of poisson
# probability distribution with different parameters
#
({
?T(n)_n
if(?T(n)<=2) par(mfrow=c(1,2), oma=c(0,0,2,0))
if(?T(n)> 2) par(mfrow=c(2,2), oma=c(0,0,2,0))
?T(nv)_c(nv)
?T(pv)_pv
?T(x)_seq(0,?T(pv),1)
?T(y)_matrix(0,nrow=?T(n),ncol=len(?T(x)))
for (_i in 1:?T(n) )  {
  ?T(y)[i,]_?mpoiss(?T(x),?T(nv)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_max(0.05,max(?T(y)))
param_encode("LAMBDA :", ?T(nv))
for (_i in 1: ?T(n) ) {
  plot(?T(x),?T(y)[i,],xlab="X VALUE",ylab="",type="h",
      main=param[i],ylim=c(0,?T(my)))
      }
mtext(side=3, line=0, outer=T,
"SHAPES OF POISSON PROBABILITY DISTRIBUTIONS")
rm(?T(n),?T(x),?T(nv),?T(pv),?T(y),?T(my))
})
END
```

```
MACRO normal
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?normplot",
         "?onormplot","?cont")
menu(item,action)
rm(item, action)
})
END




MACRO normplot(
m/?PROMPT(Mean              : )/,
s/?PROMPT(Sigma             : )/,
minx/?PROMPT(MIN X for PLOT   : )/,
maxx/?PROMPT(MAX X for PLOT   : )/)
#
# This is a Macro to plot the normal
# probability distribution
#
({
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(m)_m
  ?T(s)_s
  ?T(nx)_minx
  ?T(xx)_maxx
  ?T(x)_seq(?T(nx),?T(xx),len=100)
  ?T(y)_dnorm(?T(x),?T(m),?T(s))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
  ?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
  plot(?T(x),?T(y),xlab="X-VALUE",ylab="",
       main="SHAPE OF NORMAL DISTRIBUTION",
       sub=encode("MEAN :",?T(m),"  STD DEV :",?T(s)),
       ylim=c(0,?T(my)),type="l")
  mtext(side=2, line=1, outer=T, "PDF")
  rm(?T(x),?T(y),?T(my),?T(m),?T(s),?T(nx),?T(xx))
})
END
```

```
MACRO onormplot(
n/?PROMPT(NUMBER OF PLOT(up to 4) : )/,
m/?PROMPT(MEAN (ex) 2,3,4       : )/,
s/?PROMPT(Std Dev (ex)1,3,7     : )/,
minx/?PROMPT(MIN X for PLOT          : )/,
maxx/?PROMPT(MAX X for PLOT          : )/)
#
# This is a Macro to compare the shape of normal
# distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(m)_c(m)
?T(s)_c(s)
?T(y)_matrix(0,nrow=?T(n),ncol=51)
?T(x)_seq(minx,maxx,len=51)
for ( i in 1:?T(n) )  {
  ?T(y)[i,]_dnorm(?T(x),?T(m)[i],?T(s)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
slct_c(5,17,29,41)
sym_c("1","2","3","4")
plot(?T(x)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF NORMAL DISTRIBUTION with Different Parameters",
ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[51]))
lines(?T(x),?T(y)[1,])
for( i in 2:?T(n) ) {
  points(?T(x)[slct+3*(i-1)],?T(y)[i,(slct+3*(i-1))],pch=sym[i])
  lines(?T(x),?T(y)[i,])
  }
mu_rep(0,4)
sd_rep(0,4)
for ( i in 1:4 ) {
mu[i]_ifelse( i<=?T(n), ?T(m)[min(i,?T(n))], NA)
sd[i]_ifelse( i<=?T(n), ?T(s)[min(i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("Mean [",mu[1],mu[2],mu[3],mu[4],"]",
"  Std Dev [",sd[1],sd[2],sd[3],sd[4],"]"))
rm(?T(n),?T(x),?T(m),?T(s),?T(y),?T(my),mu,sd,slct,sym)
})
END
```

```
MACRO cuniform
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?cunifplot",
         "?ocunifplot","?cont")
menu(item,action)
rm(item, action)
})
END
```

```
MACRO cunifplot(
iv/?PROMPT(LOWER VALUE    : )/,
ev/?PROMPT(UPPER VALUE    : )/,
minx/?PROMPT(MIN X for PLOT : )/,
maxx/?PROMPT(MAX X for PLOT : )/)
#
# This is a Macro to plot the uniform
# probability distribution
#
({
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(i)_iv
  ?T(e)_ev
  ?T(nx)_minx
  ?T(xx)_maxx
  ?T(x)_seq(?T(nx),?T(xx),len=100)
  ?T(w)_(1:len(?T(x)))[?T(x)<?T(i) | ?T(x)>?T(e)]
  ?T(y)_rep(1/(?T(e)-?T(i)),100)
  ?T(y)[?T(w)]_0
#
# The probability for the range of outside of [i,e] is zero
#
  plot(?T(x),?T(y),xlab="X-VALUE",ylab="",
       main="SHAPE OF CONTINUOUS UNIFORM DISTRIBUTION",
       sub=encode("From : ",?T(i),"  To : ",?T(e)),type="l",
       err=-1)
  mtext(side=2, line=1, outer=T, "PDF")
  rm(?T(x),?T(y),?T(w),?T(i),?T(e),?T(xx),?T(nx))
})
END
```

```
MACRO ocunifplot(
n/?PROMPT(NUMBER OF PLOT(up to 4) : )/,
m/?PROMPT(LOWER LIMIT(ex)2,3,4    : )/,
s/?PROMPT(UPPER LIMIT(ex)5,7,9    : )/,
minx/?PROMPT(MIN X for PLOT           : )/,
maxx/?PROMPT(MAX X for PLOT           : )/)
#
# This is a Macro to compare the shape of uniform probability
# distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(m)_c(m)
?T(s)_c(s)
?T(y)_matrix(0,nrow=?T(n),ncol=51)
?T(x)_seq(minx,maxx,len=51)
for ( i in 1:?T(n) )  {
  ?T(y)[i,]_dunif(?T(x),?T(m)[i],?T(s)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
sym_c("1","2","3","4")
slct_c(5,17,29,41)
plot(?T(x)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF UNIFORM DISTRIBUTION with Different
Parameters", ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[51]))
lines(?T(x),?T(y)[1,])
for( i in 2:?T(n) ) {
  points(?T(x)[slct+3*( i-1)],?T(y)[i,(slct+3*( i-1))],pch=sym[i])
  lines(?T(x),?T(y)[i,])
  }
lv_rep(0,4)
uv_rep(0,4)
for ( i in 1:4 ) {
lv[i]_ifelse( i<=?T(n), ?T(m)[min( i,?T(n))], NA)
uv[i]_ifelse( i<=?T(n), ?T(s)[min( i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("Lower [",lv[1],lv[2],lv[3],lv[4],"]",
"  Upper [",uv[1],uv[2],uv[3],uv[4],"]"))
rm(?T(m),?T(s),?T(n),?T(x),?T(y),?T(my),lv,uv,slct,sym)
})
END
```

```
MACRO lognorm
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?lnormplot",
         "?olnormplot","?cont")
menu(item,action)
rm(item, action)
})
END
```

```
MACRO lnormplot(
mv/?PROMPT(Mean Value      : )/,
sv/?PROMPT(Std Dev         : )/,
maxx/?PROMPT(MAX X for PLOT  : )/)
#
# This is a Macro to plot the lognormal
# probability distribution
#
({
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(m)_mv
  ?T(s)_sv
  ?T(x)_maxx
  ?T(x)_seq(0,?T(x),len=100)
  ?T(p)_?T(x)[?T(x)>0]
  ?T(y)_dlnorm(?T(p),?T(m),?T(s))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
  if(max(?T(y))<0.05)?T(my)_0.05 else ?T(my)_max(?T(y))
  plot(?T(p),?T(y),xlab="X-VALUE",ylab="",
     main="SHAPE OF LOGNORMAL DISTRIBUTION",ylim=c(0,?T(my)),
     sub=encode("MEAN : ",?T(m),"  STD DEV : ",?T(s)),type="l",
     err=-1)
  mtext(side=2, line=1, outer=T, "PDF")
  rm(?T(x),?T(y),?T(p),?T(my),?T(m),?T(s))
})
END
```

```
MACRO olnormplot(
n/?PROMPT(NUMBER OF PLOT(up to 4) : )/,
m/?PROMPT(MEAN (ex)2,3,4        : )/,
s/?PROMPT(Std Dev (ex)1,3,7     : )/,
maxx/?PROMPT(MAX X for PLOT          : ) /)
#
# This is a Macro to compare the shape of lognormal
# probability distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(m)_c(m)
?T(s)_c(s)
?T(y)_matrix(0,nrow=?T(n),ncol=50)
?T(x)_seq(0,maxx,len=51)
?T(p)_?T(x)[?T(x)>0]
for (_i in 1:?T(n) )  {
  ?T(y)[i,]_dlnorm(?T(p),?T(m)[i],?T(s)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
sym_c("1","2","3}","4")
slct_c(5,17,29,41)
plot(?T(p)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF LOGNORMAL DISTRIBUTION with Different
Parameters", ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[51]), err=-1)
lines(?T(p),?T(y)[1,])
for( i in 2:?T(n) ) {
  points(?T(p)[slct+3*(i-1)],?T(y)[i,(slct+3*(i-1))],pch=sym[i])
  lines(?T(p),?T(y)[i,])
  }
mu_rep(0,4)
sd_rep(0,4)
for ( i in 1:4 ) {
mu[i]_ifelse( i<=?T(n), ?T(m)[min(i,?T(n))], NA)
sd[i]_ifelse( i<=?T(n), ?T(s)[min(i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("Mean [",mu[1],mu[2],mu[3],mu[4],"]",
"  Std Dev [",sd[1],sd[2],sd[3],sd[4],"]"))
rm(?T(n),?T(x),?T(p),?T(m),?T(s),?T(y),?T(my),mu,sd,slct,sym)
})
END
```

```
MACRO tdis
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?tplot",
         "?otplot","?cont")
menu( item,action)
rm( item, action)
})
END
```

```
MACRO tplot(
df/?PROMPT(Degree of Freedom  : )/,
nx/?PROMPT(MIN X for PLOT     : )/,
xx/?PROMPT(MAX X for PLOT     : )/)
#
# This is a Macro to plot the T probability distribution
#
({
  ?T(d)_df
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(x)_seq(nx,xx,len=100)
  ?T(y)_dt(?T(x),?T(d))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
  ?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
  plot(?T(x),?T(y),xlab="X-VALUE", ylab="",ylim=c(0,?T(my)),
       main="SHAPE OF T DISTRIBUTION",
       sub=encode("Degree of Freedom : ",?T(d)),type="l", err=-1)
  mtext(side=2, line=1, outer=T, "PDF")
  rm(?T(x),?T(y),?T(my),?T(d))
})
END
```

```
MACRO otplot(
n/?PROMPT(NUMBER OF PLOT(up to 4)    : )/,
m/?PROMPT(DEGREE OF FREEDOM(ex)2,3,4 : )/,
minx/?PROMPT(MIN X for PLOT            : )/,
maxx/?PROMPT(MAX X for PLOT            : )/)
#
# This is a Macro to compare the shape of t probability
# distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(m)_c(m)
?T(y)_matrix(0,nrow=?T(n),ncol=51)
?T(x)_seq(minx,maxx,len=51)
for ( i in 1:?T(n) )  {
  ?T(y)[i,]_dt(?T(x),?T(m)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
sym_c("1","2","3","4")
slct_c(5,17,29,41)
plot(?T(x)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF T DISTRIBUTION with Different Parameters",
ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[51]))
lines(?T(x),?T(y)[1,])
for( i in 2:?T(n) ) {
  points(?T(x)[slct+3*(i-1)],?T(y)[i,(slct+3*(i-1))],pch=sym[i])
  lines(?T(x),?T(y)[i,])
  }
mu_rep(0,4)
for ( i in 1:4 ) {
mu[i]_ifelse( i<=?T(n), ?T(m)[min(i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("Degree of Freedom [",mu[1],mu[2],mu[3],mu[4],"]"))
rm(?T(n),?T(x),?T(m),?T(y),?T(my),mu,slct,sym)
})
END
```

```
MACRO expon
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?explot",
         "?oexplot","?cont")
menu(item,action)
rm(item, action)
})
END
```

```
MACRO explot(
r/?PROMPT(LAMBDA ( L > 0) : )/,
x/?PROMPT(MAX X for PLOT  : )/)
# This is a Macro to plot the exponential
# probability distribution
#
({
   par(mfrow=c(1,1), oma=c(0,2,0,0))
   ?T(r)_r
   ?T(x)_x
   ?T(x)_seq(0,?T(x),len=100)
   ?T(y)_?dex1(?T(x),?T(r))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
   ?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
   plot(?T(x),?T(y),xlab="X-VALUE",ylab="",ylim=c(0,?T(my)),
        main="SHAPE OF EXPONENTIAL DISTRIBUTION",
        sub=encode("LAMBDA : ", ?T(r)),type="l")
   mtext(side=2, line=1, outer=T, "PDF")
   rm(?T(x),?T(y),?T(my),?T(r))
})
END
```

```
MACRO oexplot(
n/?PROMPT(NUMBER OF PLOT(up to 4) : )/,
r/?PROMPT(LAMBDA (Ex; 2,4,8 (>0)) : )/,
maxx/?PROMPT(MAX X for PLOT         : )/)
#
# This is a Macro to compare the shape of expontential
# probability distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(r)_c(r)
?T(y)_matrix(0,nrow=?T(n),ncol=51)
?T(x)_seq(0,maxx,len=51)
for ( i in 1:?T(n) )  {
  ?T(y)[i,]_?dex1(?T(x),?T(r)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
sym_c("1","2","3","4")
slct_c(5,17,29,41)
plot(?T(x)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF EXPONENTIAL DISTRIBUTION with Different
Parameters", ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[51]), err=-1)
lines(?T(x),?T(y)[1,])
for( i in 2:?T(n) ) {
  points(?T(x)[slct+3*(i-1)],?T(y)[i,(slct+3*(i-1))],pch=sym[i])
  lines(?T(x),?T(y)[i,])
  }
lm_rep(0,4)
for ( i in 1:4 ) {
lm[i]_ifelse( i<=?T(n), ?T(r)[min(i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("Lambda [",lm[1],lm[2],lm[3],lm[4],"]"))
rm(?T(n),?T(x),?T(r),?T(y),?T(my),lm,slct,sym)
})
END
```

```
MACRO gammaset
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?gamplot",
         "?ogamplot","?cont")
menu(item,action)
})
END
```

```
MACRO gamplot(
a/?PROMPT(ALPHA VALUE( A > 0 ) : )/,
b/?PROMPT(BETA VALUE( B > 0 )  : )/,
maxx/?PROMPT(MAX X for PLOT      : )/)
#
# This is a Macro to plot the gamma
# probability distribution
#
({
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(a)_a
  ?T(b)_b
  ?T(x)_seq(0,maxx,len=100)
#
# Exclude the point x = 0
#
  ?T(x)_?T(x)[?T(x)>0]
  ?T(y)_?gamfun1(?T(x),?T(a),?T(b))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
  ?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
  plot(?T(x),?T(y),xlab="X-VALUE",ylab="",
     main="SHAPE OF GAMMA DISTRIBUTION",ylim=c(0,?T(my)),
     sub=encode("Alpha : ",?T(a),"  Beta : ",?T(b)),type="l")
  mtext(side=2, line=1, outer=T, "PDF")
  rm(?T(x),?T(y),?T(my),?T(a),?T(b))
})
END
```

```
MACRO ogamplot(
n/?PROMPT(NUMBER OF PLOT(up to 4)      : )/,
m/?PROMPT(ALPHA(positiv);ex,2,1,2  : )/,
s/?PROMPT(BETA(positive);ex,.3,1,2 : )/,
maxx/?PROMPT(MAX X for PLOT            : )/)
#
# This is a Macro to compare the shape of gamma probability
# distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(m)_c(m)
?T(s)_c(s)
?T(y)_matrix(0,nrow=?T(n),ncol=50)
?T(x)_seq(0,maxx,len=51)
?T(x)_?T(x)[?T(x)>0]
 for ( i in 1:?T(n) )  {
  ?T(y)[i,]_?gamfun1(?T(x),?T(m)[i],?T(s)[i])
   }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
sym_c("1","2","3","4")
slct_c(5,17,29,41)
plot(?T(x)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF GAMMA DISTRIBUTION with Different Parameters",
ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[50]))
lines(?T(x),?T(y)[1,])
for( i in 2:?T(n) ) {
  points(?T(x)[slct+3*(i-1)],?T(y)[i,(slct+3*(i-1))],pch=sym[i])
  lines(?T(x),?T(y)[i,])
   }
al_rep(0,4)
be_rep(0,4)
 for (  i in 1:4 ) {
al[i]_ifelse( i<=?T(n), ?T(m)[min(i,?T(n))], NA)
be[i]_ifelse( i<=?T(n), ?T(s)[min(i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("Alpha [",al[1],al[2],al[3],al[4],"]",
"  Beta [",be[1],be[2],be[3],be[4],"]"))
rm(?T(n),?T(x),?T(m),?T(s),?T(y),?T(my),al,be,slct,sym)
})
END
```

```
MACRO weibull
({
 item_c("SHAPE OF DISTRIBUTION",
        "SHAPE COMPARISON",
        "RETURN TO PREVIOUS STEP")
action_c("?weibplot",
         "?oweibplot","?cont")
menu(item,action)
})
END




MACRO weibplot(
a/?PROMPT(Shape Parameter ( > 0 ) : )/,
b/?PROMPT(Scale Parameter ( > 0 ) : )/,
mx/?PROMPT(MAX X for PLOT          : )/)
#
# This is a Macro to plot the weibull
# probability distribution
#
({
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(a)_a
  ?T(b)_b
  ?T(x)_seq(0,mx,len=100)
  ?T(x)_?T(x)[?T(x)>0]
  ?T(y)_?dweib1(?T(x),?T(a),?T(b))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
  ?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
  plot(?T(x),?T(y),xlab="X-VALUE",ylab="",ylim=c(0,?T(my)),
       main="SHAPE OF WEIBULL DISTRIBUTION", err=-1,
       sub=encode("ALPHA : ",?T(a),"  BETA : ",?T(b)),type="l")
  mtext(side=2, line=1, outer=T, "PDF")
  rm(?T(x),?T(y),?T(my),?T(a),?T(b))
})
END
```

```
MACRO oweibplot(
n/?PROMPT(NUMBER OF PLOT(up to 4)     : )/,
m/?PROMPT(Shape Parameter  ex;2,1,2 : )/,
s/?PROMPT(Scale Parameter  ex;3,5,1 : )/,
maxx/?PROMPT(Max X for PLOT              : )/)
#
# This is a Macro to compare the shape of weibull probability
# distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(m)_c(m)
?T(s)_c(s)
?T(y)_matrix(0,nrow=?T(n),ncol=100)
?T(x)_seq(0,maxx,len=101)
#
# Exclude the point x=0
#
?T(x)_?T(x)[?T(x)>0]
 for ( i in 1:?T(n) )  {
   ?T(y)[i,]_?dweib1(?T(x),?T(m)[i],?T(s)[i])
   }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
sym_c("1","2","3","4")
slct_c(5,34,58,82)
plot(?T(x)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF WEIBULL DISTRIBUTION with Different
Parameters", ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[100]), err=-1)
lines(?T(x),?T(y)[1,])
 for( i in 2:?T(n) ) {
   points(?T(x)[slct+3*(i-1)],?T(y)[i,(slct+3*(i-1))],pch=sym[i])
   lines(?T(x),?T(y)[i,])
   }
al_rep(0,4)
be_rep(0,4)
 for ( i in 1:4 ) {
al[i]_ifelse( i<=?T(n), ?T(m)[min(i,?T(n))], NA)
be[i]_ifelse( i<=?T(n), ?T(s)[min(i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("Shape P [",al[1],al[2],al[3],al[4],"]",
"  Scale P [",be[1],be[2],be[3],be[4],"]"))
rm(?T(n),?T(x),?T(m),?T(s),?T(y),?T(my),al,be,slct,sym)
})
END
```

```
MACRO betaset
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?betaplot",
         "?obetaplot","?cont")
menu( item,action)
rm( item, action)
})
END
```

```
MACRO betaplot(
av/?PROMPT(ALPHA VALUE ( A > 0  ) : )/,
bv/?PROMPT(BETA  VALUE ( B > 0 )  : ) /)
#
# This is a Macro to plot the beta
# probability distribution
#
({
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(a)_av
  ?T(b)_bv
  ?T(x)_seq(0,1,len=102)
#
# Exclude the point x=0 and x=1
#
  ?T(x)_?T(x)[?T(x)>0 & ?T(x)<1]
  ?T(y)_dbeta(?T(x),?T(a),?T(b))
  plot(?T(x),?T(y),xlab="X-VALUE",ylab="",type="l",
       main="SHAPE OF BETA DISTRIBUTION",
       sub=encode("ALPHA : ",?T(a),"  BETA : ",?T(b)))
  mtext(side=2, line=1, outer=T, "PDF")
  rm(?T(x),?T(y),?T(a),?T(b))
})
END
```

```
MACRO obetaplot(
n/?PROMPT(NUMBER OF PLOT(up to 4)       : )/,
m/?PROMPT(ALPHA(positive);ex,2,1,2  : )/,
s/?PROMPT(BETA (positive);ex,.3,1,2 : )/,
maxx/?PROMPT(MAX X (less than 1)          : )/)
#
# This is a Macro to compare the shape of beta probability
# distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(m)_c(m)
?T(s)_c(s)
?T(y)_matrix(0,nrow=?T(n),ncol=50)
?T(x)_seq(0,maxx,len=51)
#
# Exclude the point x=0
#
?T(x)_?T(x)[?T(x)>0]
 for ( i in 1:?T(n) )  {
   ?T(y)[i,]_dbeta(?T(x),?T(m)[i],?T(s)[i])
   }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
sym_c("1","2","3","4")
slct_c(5,17,29,41)
plot(?T(x)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF BETA DISTRIBUTION with Different
Parameters", ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[50]), err=-1)
lines(?T(x),?T(y)[1,])
for( i in 2:?T(n) ) {
  points(?T(x)[slct+3*(i-1)],?T(y)[i,(slct+3*(i-1))],pch=sym[i])
  lines(?T(x),?T(y)[i,])
  }
al_rep(0,4)
be_rep(0,4)
for ( i in 1:4 ) {
al[i]_ifelse( i<=?T(n), ?T(m)[min(i,?T(n))], NA)
be[i]_ifelse( i<=?T(n), ?T(s)[min(i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("ALPHA [",al[1],al[2],al[3],al[4],"]",
"  BETA [",be[1],be[2],be[3],be[4],"]"))
rm(?T(n),?T(m),?T(s),?T(x),?T(y),al,be,sym,slct)
})
END
```

```
MACRO chisquare
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?chisqplot",
       "?ochisqplot","?cont")
menu( item,action)
rm( item, action)
})
END
```

```
MACRO chisqplot(
m/?PROMPT(DEGREE OF FREEDOM  : )/,
maxx/?PROMPT(MAX X for PLOT     : )/)
#
# This is a Macro to plot the chisquareetric
# probability distribution
#
({
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(m)_m
  ?T(x)_seq(0,maxx,len=100)
#
# Exclude the point x=0
#
  ?T(x)_?T(x)[?T(x)>0]
  ?T(y)_dchisq(?T(x),?T(m))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
  ?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
  plot(?T(x),?T(y),xlab="X-VALUE",ylab="",
     main="SHAPE OF CHISQUARE DISTRIBUTION",ylim=c(0,?T(my)),
     sub=encode("DEGREE OF FREEDOM : ", ?T(m)),type="l")
  mtext(side=2, line=1, outer=T, "PDF")
  rm(?T(x),?T(y),?T(my),?T(m))
})
END
```

```
MACRO ochisqplot(
n/?PROMPT(NUMBER OF PLOT(up to 4)      : )/,
m/?PROMPT(DEGREE OF FREEDOM(ex)2,3,4 : )/,
maxx/?PROMPT(MAX X for PLOT              :  )/)
#
# This is a Macro to compare the shape of chisquare
# probability distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(m)_c(m)
?T(y)_matrix(0,nrow=?T(n),ncol=50)
?T(x)_seq(0,maxx,len=51)
#
# Exclude the point x=0
#
?T(x)_?T(x)[?T(x)>0]
for ( i in 1:?T(n) )  {
  ?T(y)[i,]_dchisq(?T(x),?T(m)[i])
  }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
sym_c("1","2","3","4")
slct_c(5,17,29,41)
plot(?T(x)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF CHISQUARE DISTRIBUTION with Different
Parameters", ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[50]))
lines(?T(x),?T(y)[1,])
for( i in 2:?T(n) ) {
  points(?T(x)[slct+3*(i-1)],?T(y)[i,(slct+3*(i-1))],pch=sym[i])
  lines(?T(x),?T(y)[i,])
  }
mu_rep(0,4)
for ( i in 1:4 ) {
mu[i]_ifelse(i<=?T(n), ?T(m)[min(i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("Degree of Freedom [",mu[1],mu[2],mu[3],mu[4],"]"))
rm(?T(n),?T(x),?T(m),?T(y),?T(my),mu,slct,sym)
})
END
```

```
MACRO fdis
({
item_c("SHAPE OF DISTRIBUTION",
       "SHAPE COMPARISON",
       "RETURN TO PREVIOUS STEP")
action_c("?fplot",
         "?ofplot","?cont")
menu(item,action)
rm(item, action)
})
END
```

```
MACRO fplot(
dfn/?PROMPT(D.F 1                : )/,
dfd/?PROMPT(D.F 2                : )/,
maxx/?PROMPT(MAX X for PLOT       : )/)
#
# This is a Macro to plot the F probability distribution
({
  par(mfrow=c(1,1), oma=c(0,2,0,0))
  ?T(n)_dfn
  ?T(d)_dfd
  ?T(x)_seq(0,maxx,len=102)
#
# Exclude the point x=0
#
  ?T(x)_?T(x)[?T(x)>0]
  ?T(y)_df(?T(x),?T(n),?T(d))
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
  ?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
  plot(?T(x),?T(y),xlab="X-VALUE",ylab="",type="l",
     main="SHAPE OF F DISTRIBUTION",ylim=c(0,?T(my)),
     sub=encode("DF numerator:",?T(n),"DF denominator:",?T(d)))
  mtext(side=2, line=1, outer=T, "PDF")
  rm(?T(x),?T(y),?T(my),?T(n),?T(d))
})
END
```

```
MACRO ofplot(
n/?PROMPT(NUMBER OF PLOT(up to 4)    : )/,
dfn/?PROMPT(D.F 1(ex)3,5,10          : )/,
dfd/?PROMPT(D.F 2(ex)5,9,13          : )/,
maxx/?PROMPT(MAX X for PLOT             : )/)
#
# This is a Macro to compare the shape of F probability
# distribution with different parameters
#
({
par(mfrow=c(1,1), oma=c(2,2,0,0))
?T(n)_n
?T(m)_c(dfn)
?T(s)_c(dfd)
?T(y)_matrix(0,nrow=?T(n),ncol=50)
?T(x)_seq(0,maxx,len=51)
#
# Exclude the point x=0
#
?T(x)_?T(x)[?T(x)>0]
 for ( i in 1:?T(n) )  {
   ?T(y)[i,]_df(?T(x),?T(m)[i],?T(s)[i])
   }
#
# To prevent too small y-axis, set the y-axis limit at least 0.05
#
?T(my)_ifelse(max(?T(y))<0.05,0.05,max(?T(y)))
slct_c(5,17,29,41)
sym_c("1","2","3","4")
plot(?T(x)[slct],?T(y)[1,slct],pch=sym[1],xlab="X VALUE",ylab="",
type="p", main="SHAPE OF F DISTRIBUTION with Different Parameters",
ylim=c(0,?T(my)),xlim=c(?T(x)[1],?T(x)[50]))
lines(?T(x),?T(y)[1,])
 for( i in 2:?T(n) ) {
   points(?T(x)[slct+3*(i-1)],?T(y)[i,(slct+3*(i-1))],pch=sym[i])
   lines(?T(x),?T(y)[i,])
   }
d1_rep(0,4)
d2_rep(0,4)
 for ( i in 1:4 ) {
d1[i]_ifelse( i<=?T(n), ?T(m)[min(i,?T(n))], NA)
d2[i]_ifelse( i<=?T(n), ?T(s)[min(i,?T(n))], NA)
}
mtext(side=2, line=1, outer=T, "PDF")
mtext(side=1, line=1, outer=T,
encode("DF 1 [",d1[1],d1[2],d1[3],d1[4],"]",
"  DF 2 [",d2[1],d2[2],d2[3],d2[4],"]"))
rm(?T(n),?T(x),?T(m),?T(s),?T(y),?T(my),d1,d2,slct,sym)
})
END
```

```
MACRO bvd
#
# This is a macro to provide information about continuous
# bivariate distribution.
#
({
  item_c("NORMAL BIVARIATE DISTRIBUTION",
         "INDEPENDENT GAMMA BIVARIATE DISTRIBUTION",
         "QUIT")
  action_c("?binormal","?bigammaset","q")
  menu( item , action)
})
END
```

```
MACRO binormal(
x/?PROMPT(X value    : )/,
mx/?PROMPT(MU X       : )/,
sx/?PROMPT(Sigma X    : )/,
y/?PROMPT(Y value    : )/,
my/?PROMPT(MU Y       : )/,
sy/?PROMPT(Sigma Y    : )/)
({
item_c("SHAPE OF BIVARIATE PDF",
       "SHAPE OF INDEPENDENT BIVARIATE CDF",
       "PDF",
       "INDEPENDENT CDF",
       "GENERATE RANDOM DEVIATES",
       "RETURN TO PREVIOUS STEP")
action_c("?binorm",
         "?bicnorm",
         "?dbinorm",
         "pnorm(x,mx,sx) *
              pnorm(y,my,sy)",
         "?rbinorm",
         "?bvd")
menu( item,action)
rm( item, action)
})
END
```

```
MACRO binorm(
mx/?PROMPT(MU X(with smaller S.D) : )/,
sx/?PROMPT(Sigma X                : )/,
my/?PROMPT(MU Y                   : )/,
sy/?PROMPT(Sigma Y                : )/,
r/?PROMPT(Correlation(-1< R <1)  : )/,
int/?PROMPT(Number of Interval     : )/)
#
# This is a Macro to plot PDF Surface for both Independent and
# Dependent Bivariate Normal Probability
#
({
  par(mfrow=c(1,2), oma=c(3,0,2,0))
  ?T(mx)_mx
  ?T(sx)_sx
  ?T(my)_my
  ?T(sy)_sy
  ?T(r)_r
  ?T(i)_int
  ?T(x)_seq((?T(mx)-2*?T(sx)),(?T(mx)+2*?T(sx)),len=?T(i))
# To set the range of X
  ?T(y)_seq((?T(my)-2*?T(sy)),(?T(my)+2*?T(sy)),len=?T(i))
# To set the range of Y
  ?T(xr)_rep(?T(x),rep(?T(i),?T(i)))
  ?T(yr)_rep(?T(y),?T(i))
  ?T(z)_?dbinorm1(?T(mx),?T(sx),?T(my),?T(sy),?T(r),?T(xr),?T(yr))
  ?T(z)_matrix(?T(z),nrow=?T(i),byrow=TRUE)
  ?T(zp)_?T(z)/max(?T(z))
# To get larger value for Z, devided by maximum Z value.
# This will make sure that largest Z is 1.
  ?T(rx)_max(?T(x))-min(?T(x))
  ?T(ry)_max(?T(y))-min(?T(y))
  ?T(ar)_?T(rx)/?T(ry)
# To get the ratio of X and Y, (maxx-minx)/(maxy-miny)
  persp(?T(zp),,?T(ar))
# View point is default value c(-6,-8,5), if you want to change
# the view point, you can use other value c(x,y,z).
  title(sub="Range of Y(L) & X(R) : Mu +/- 2*S")
  ?T(lm)_(max(?T(rx),?T(ry)))/2
  contour(?T(x),?T(y),?T(z),xlim=c((?T(mx)-?T(lm)),(?T(mx)+?T(lm))),
     ylim=c((?T(my)-?T(lm)),(?T(my)+?T(lm))),nint=3)
# To get the same distance of X and Y axis, use xlim and ylim
  title(xlab="x value",ylab="y value")
  mtext(side=3, line=1, outer=T,
     "PDF Surface & Contour Plot for Bivarate Normal Distribution")
  mtext(side=1, line=1, outer=T,
     encode("M1 =",?T(mx)," S1 =",?T(sx),"  M2 =",?T(my),
          " S2 =",?T(sy),"  R =",?T(r)))
  rm(?T(mx),?T(sx),?T(my),?T(sy),?T(r),?T(i),?T(z),?T(zp),
     ?T(xr),?T(yr),?T(x),?T(y),?T(lm),?T(ar),?T(rx),?T(ry))
})
END
```

```
MACRO bicnorm(
mx/?PROMPT(MU X(with smaller S.D) : )/,
sx/?PROMPT(Sigma X                : )/,
my/?PROMPT(MU Y                   : )/,
sy/?PROMPT(Sigma Y                : )/,
int/?PROMPT(Number of Interval     : )/)
#
# This is a Macro to plot CDF Surface for Independent
# Bivariate Normal Probability
#
({
  par(mfrow=c(1,1), oma=c(3,0,2,0))
  ?T(mx)_mx
  ?T(sx)_sx
  ?T(my)_my
  ?T(sy)_sy
  ?T(i)_int
  ?T(x)_seq((?T(mx)-3*?T(sx)),(?T(mx)+3*?T(sx)),len=?T(i))
# To set the range of X
  minx_min(?T(x))   # minimum of x
  maxx_max(?T(x))   # maximum of x
  ?T(y)_seq((?T(my)-3*?T(sy)),(?T(my)+3*?T(sy)),len=?T(i))
# To set the range of Y
  miny_min(?T(y))   # minimum of y
  maxy_max(?T(y))   # maximum of y
  ?T(zx)_pnorm(?T(x),?T(mx),?T(sx))
  ?T(zy)_pnorm(?T(y),?T(my),?T(sy))
  ?T(zx)_matrix(?T(zx),nrow=?T(i))
  ?T(zy)_matrix(?T(zy),ncol=?T(i))
  ?T(z)_?T(zx)%*?T(zy)
  ?T(zp)_?T(z)/max(?T(z))
# To get larger value for Z, devided by maximum Z value.
# This will make sure that largest Z is 1.
  ?T(ar)_(maxx-minx)/(maxy-miny)
# To get the ratio of X and Y, (maxx-minx)/(maxy-miny)
  persp(?T(zp),,?T(ar))
# View point is default value c(-6,-8,5), if you want to change
# the view point, you can use other value c(x,y,z).
  mtext(side=3, line=1, outer=T,
      "CDF Surface for Bivariate Independent Normal Distribution")
  mtext(side=1, line=0, outer=T,
      encode("M1=",?T(mx),"S1=",?T(sx),"M2=",?T(my),"S2=",?T(sy)))
  mtext(side=1, line=2, outer=T,
    encode("Y(L) : [",miny,maxy,"]   X(R) : [",minx,maxx,"]"))
  rm(?T(mx),?T(sx),?T(my),?T(sy),?T(i),?T(z),?T(zp),
     ?T(x),?T(y),?T(ar),minx,maxx,miny,maxy)
})
END
```

```
MACRO bigammaset(
ax/?PROMPT(Alpha X   : )/,
bx/?PROMPT(Beta X    : )/,
ay/?PROMPT(Alpha Y   : )/,
by/?PROMPT(Beta Y    : )/,
x/?PROMPT(X value   : )/,
y/?PROMPT(Y value   : )/)
({
item_c("SHAPE OF INDEPENDENT BIVARIATE PDF",
       "SHAPE OF INDEPENDENT STANDARD BIVARIATE CDF",
       "INDEPENDENT PDF",
       "INDEPENDENT STANDARD CDF",
       "GENERATE RANDOM DEVIATES",
       "RETURN TO PREVIOUS STEP")
action_c("?bigamma",
         "?bicgamma",
         "?gamfun1(x,ax,bx) *
               ?gamfun1(y,ay,by)",
         "pgamma(x,ax) *
               pgamma(y,ay)",
         "?rbigamma",
         "?bvd")
menu(item,action)
rm(item, action)
})
END
```

```
MACRO bigamma(
ax/?PROMPT(Alpha X     : )/,
bx/?PROMPT(Beta X      : )/,
ay/?PROMPT(Alpha Y     : )/,
by/?PROMPT(Beta Y      : )/,
m/?PROMPT(Maximum range of X and Y : )/,
int/?PROMPT(Number of Interval        : )/)
#
# This is Macro to plot the PDF Surface for
# Independent Bivariate Gamma Probability
#
({
  par(mfrow=c(1,2), oma=c(3,0,2,0))
  ?T(ax)_ax
  ?T(bx)_bx
  ?T(ay)_ay
  ?T(by)_by
  ?T(m)_m
  ?T(i)_int
  ?T(x)_seq(0,?T(m),len=?T(i))
# To set the range of X
  ?T(y)_seq(0,?T(m),len=?T(i))
# To set the range of Y
  ?T(zx)_c(0,?gamfun1(?T(x)[2:?T(i)],?T(ax),?T(bx)))
  ?T(zy)_c(0,?gamfun1(?T(y)[2:?T(i)],?T(ay),?T(by)))
  ?T(zx)_matrix(?T(zx),nrow=?T(i))
  ?T(zy)_matrix(?T(zy),ncol=?T(i))
  ?T(z)_?T(zx)%*%?T(zy)
  ?T(zp)_?T(z)/max(?T(z))
# To get larger value for Z, devided by maximum Z value.
# This will make sure that largest Z is 1.
  persp(?T(zp))
# View point is default value c(-6,-8,5), if you want to change
# the view point, you can use other value c(x,y,z).
  contour(?T(x),?T(y),?T(z),nint=3,xlab="x value",ylab="y value")
  mtext(side=3, line=1, outer=T,
     "PDF Surface & Contour Plot for Bivariate Gamma Distribution")
  mtext(side=1, line=1, outer=T,
     encode("A1=",?T(ax),"B1=",?T(bx),"A2=",?T(ay),"B2=",?T(by)))
  rm(?T(ax),?T(bx),?T(ay),?T(by),?T(m),?T(i),?T(z),?T(zp),
     ?T(x),?T(y),?T(zx),?T(zy))
})
END
```

```
MACRO bicgamma(
ax/?PROMPT(Alpha X     : )/,
ay/?PROMPT(Alpha Y     : )/,
m/?PROMPT(Maximum range of X and Y : )/,
int/?PROMPT(Number of Interval       : )/)
#
# This is Macro to plot the CDF Surface for
# Independent Standard Gamma Probability
({
  par(mfrow=c(1,1), oma=c(3,0,2,0))
  ?T(ax)_ax
  ?T(ay)_ay
  ?T(m)_m
  ?T(i)_int
  ?T(x)_seq(0,?T(m),len=?T(i))
# To set the range of X
  ?T(y)_seq(0,?T(m),len=?T(i))
# To set the range of Y
  ?T(zx)_c(0,pgamma(?T(x)[2:?T(i)],?T(ax)))
  ?T(zy)_c(0,pgamma(?T(y)[2:?T(i)],?T(ay)))
  ?T(zx)_matrix(?T(zx),nrow=?T(i))
# To make ?T(zx) m by 1 matrix
  ?T(zy)_matrix(?T(zy),ncol=?T(i))
# To make ?T(zy) 1 by m matrix
  ?T(z)_?T(zx)%*?T(zy)
  ?T(zp)_?T(z)/max(?T(z))
# To get larger value for Z, devided by maximum Z value.
# This will make sure that largest Z is 1.
  persp(?T(zp))
# View point is default value c(-6,-8,5), if you want to change
# the view point, you can use other value c(x,y,z).
  mtext(side=3, line=1, outer=T,
      "CDF Surface for Bivariate Independent Gamma Distribution")
  mtext(side=1, line=0, outer=T,
      encode("Alpha 1 =",?T(ax),"   Alpha 2 =",?T(ay)))
  mtext(side=1, line=2, outer=T,
    encode("Range of X(R) & Y(L) : [ 0 ,",?T(m),"]"))
  rm(?T(ax),?T(ay),?T(m),?T(i),?T(z),?T(x),?T(y),?T(zx),
     ?T(zy),?T(zp))
})
END
```

Appendix B: Macros for Maximum Likelihood Estimation

Table of Contents

```
MACRO mle
({
   item_c("Discrete   Distribution",
          "Continuous Distribution",
          "Quit")
   action_c("?mledisc",
            "?mlecont",
            "q")
   menu( item , action)
   rm( item, action)
})
END




MACRO mledisc
({
   item_c("Bernoulli Distribution",
          "Poisson   Distribution",
          "Geometric Distribution",
          "ReyLamb   Distribution",
          "Uniform   Distribution",
          "Go to previous step")
   action_c("?mleber",
            "?mlepos",
            "?mlegeo",
            "?mlerey",
            "?mledunif",
            "?mle")
   menu( item , action)
   rm( item, action)
})
END




MACRO mleber
({
   item_c("Sample size one",
          "Sample size two",
          "Go to previous step")
   action_c("?mleber1",
            "?mleber2",
            "?mledisc")
   menu( item , action)
   rm( item, action)
})
END
```

```
MACRO mleber1(x/?PROMPT(Random Sample 0 or 1 :)/,
p/?PROMPT(Estimated P( X = 1 ) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Bernoulli distribution.(Sample size one)
#
({
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    ?T(r)_x
    ?T(p)_p
    ?T(x)_c(0,1)
    ?T(y)_?mber(?T(x),?T(p))
    ?T(iy)_?mber(?T(r),?T(p))
    plot(?T(x),?T(y),
       main="PMF for Bernoulli Distribution",
       sub=encode("For Given X:",?T(r)," When P =",?T(p),"PMF =",
       ?T(iy)),type="h",xlab="X-Value", ylab="",
       xlim=c(-0.5,1.5),ylim=c(0,1),lab=c(2,5))
    ?T(py)_?mber(?T(r),?T(p))
    arrows(-.5,?T(py),-1,?T(py))
    points(?T(r),?T(py),mark=0)
    mtext(side=2, line=1, outer=T, "PMF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for P, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp)_seq(0,1,len=51)
    ?T(lp)_?mber(?T(r),?T(pp))
    ?T(ml)_max(?T(lp))
    ?T(mp)_?T(pp)[(1:len(?T(lp)))[?T(lp)==?T(ml)]]
    par(mfrow=c(1,1), oma=c(0,2,0,0))
    plot(?T(pp),?T(lp),type="l",
      main="Maximum Likelihood for Bernoulli Distribution",
      xlab="P-value",ylab="",xlim=c(-.2,1.2),ylim=c(0,1.1*?T(ml)),
      sub=encode("For Given X:",?T(r),"When P =",?T(mp),
      " L(P) =",?T(ml)))
    mtext(side=2, line=1, outer=T, "MLE")
    lines(c(?T(p),?T(p),-.2),c(0,?T(py),?T(py)))
    arrows(0,?T(ml),-.2,?T(ml))
    arrows(?T(mp),?T(ml)/5,?T(mp),0)
    rm(?T(x),?T(p),?T(y),?T(r),?T(pp),?T(lp),?T(py),?T(iy),
       ?T(ml),?T(mp))
 })
END
```

```
MACRO mleber2(x/?PROMPT(Random Sample X; 0 or 1      :)/,
y/?PROMPT(Random Sample Y; 0 or 1      :)/,
p/?PROMPT(Estimated P( X or Y = 1 )    :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Bernoulli distribution(Sample size two).
#
({
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(p)_p
    ?T(xx)_c(0,1)
    ?T(yy)_c(0,1)
    ?T(z)_matrix(?mber(?T(xx),?T(p)),2,1) %*
                matrix(?mber(?T(yy),?T(p)),1,2)
    ?T(iy)_?mber(?T(x),?T(p))*?mber(?T(y),?T(p))
    plot(rep(?T(xx),c(2,2)),rep(?T(yy),2),
         main="PMF for Bernoulli Distribution",
         sub=encode("For Given  X:",?T(x)," Y:",?T(y)," When P =",
         ?T(p),"PMF =",?T(iy)),type="n",lab=c(2,2),
         xlab="X-VALUE", ylab="",xlim=c(-.5,1.5),ylim=c(-.5,1.5))
    points(?T(x),?T(y),mark=1)
    lines(c(0,0),c(0,?T(z)[1,1]))
    lines(c(0,0),c(1,1+?T(z)[1,2]))
    lines(c(1,1),c(0,?T(z)[2,1]))
    lines(c(1,1),c(1,1+?T(z)[2,2]))
    mtext(side=2, line=1, outer=T, "Y Value")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for P, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp)_seq(0,1,len=51)
    ?T(lp)_?mber(?T(x),?T(pp))*?mber(?T(y),?T(pp))
    ?T(ml)_max(?T(lp))
    ?T(mp)_?T(pp)[(1:len(?T(lp)))[?T(lp)==?T(ml)]]
    par(mfrow=c(1,1), oma=c(0,2,0,0))
    plot(?T(pp),?T(lp),type="l",
      main="Maximum Likelihood for Bernoulli Distribution",
      xlab="P-value",ylab="",xlim=c(-.2,1.2),ylim=c(0,1.1*?T(ml)),
      sub=encode("For Given X:",?T(x)," Y:",?T(y)," When P =",?T(mp),
      " L(P) =",?T(ml)))
    lines(c(?T(p),?T(p),-.2),c(0,?T(iy),?T(iy)))
    arrows(0,?T(ml),-.2,?T(ml))
    arrows(?T(mp),?T(ml)/5,?T(mp),0)
    mtext(side=2, line=1, outer=T, "MLE")
    rm(?T(x),?T(p),?T(y),?T(pp),?T(lp),?T(iy),?T(xx),?T(yy),
       ?T(z),?T(ml),?T(mp))
 })
END
```

```
MACRO mlepos
({
  item_c("Sample size one",
         "Sample size two",
         "Go to previous step")
  action_c("?mlepos1",
           "?mlepos2",
           "?mledisc")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO mlepos1(
x/?PROMPT(Random Sample  (Positive Integer Up to 30) :)/,
r/?PROMPT(Estimated Lambda ( greater than 0 )        :)/)
#
# This is a Macro to plot the maximum likelihood estimator
# for Poisson distribution.(Sample size one)
#
({
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    ?T(x)_x
    ?T(r)_r
    ?T(px)_0:30
    ?T(y)_?mpoiss(?T(px),?T(r))
    ?T(py)_?mpoiss(?T(x),?T(r))
    ?T(my)_max(0.05,max(?T(y)))
    plot(?T(px),?T(y),
         main="PMF for Poisson Distribution",
         sub=encode("For Given X:",?T(x)," When Lambda =",?T(r),
         "PMF =",?T(py)),type="h",xlim=c(-5,30),
         xlab="X-VALUE", ylab="",ylim=c(0,?T(my)*6/5))
    arrows(?T(x),?T(py)+?T(my)/5,?T(x),?T(py)+?T(my)/20)
    arrows(-1,?T(py),-5,?T(py))
    mtext(side=2, line=1, outer=T, "PMF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for Lambda, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(rr)_seq(0,40,len=81)
    ?T(rr)_?T(rr)[?T(rr)>0]
    ?T(lr)_?mpoiss(rep(?T(x),len(?T(rr))),?T(rr))
    ?T(ml)_?mpoiss(?T(x),?T(x))
    par(mfrow=c(1,1), oma=c(0,2,0,0))
    plot(?T(rr),?T(lr),type="l",
       main="Maximum Likelihood for Poisson Distribution",
       xlab="Lambda",ylab="",
       sub=encode("For Given X:",?T(x)," When Lambda =",?T(x),
       " L(L) =",?T(ml)))
    lines(c(?T(r),?T(r),0),c(0,?T(py),?T(py)))
    arrows(?T(x),?T(ml),0,?T(ml))
    arrows(?T(x),?T(ml),?T(x),0)
    mtext(side=2, line=1, outer=T, "MLE")
    rm(?T(x),?T(r),?T(y),?T(rr),?T(lr),?T(py),?T(my),
       ?T(px),?T(ml))
 })
END
```

```
MACRO mlepos2(
x/?PROMPT(Sample X (Positive Integer Up to 30) :)/,
y/?PROMPT(Sample Y (Positive Integer Up to 30) :)/,
r/?PROMPT(Estimated Lambda  ( greater than 0 ) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Poisson distribution(Sample size two).
#
({
    par(mfrow=c(1,1), oma=c(5,0,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(r)_r
    ?T(px)_max(0,?T(x)-5)
    ?T(px)_seq(?T(px),?T(px)+10)
    ?T(py)_max(0,?T(y)-5)
    ?T(py)_seq(?T(py),?T(py)+10)
    ?T(zx)_matrix(?mpoiss(?T(px),?T(r)),11,1)
    ?T(zy)_matrix(?mpoiss(?T(py),?T(r)),1,11)
    ?T(z)_?T(zx)%*?T(zy)
    ?T(z)_?T(z)/max(?T(z))
    ?T(yy)_?mpoiss(?T(x),?T(r))*?mpoiss(?T(y),?T(r))
    persp(?T(z))
    title(main="PMF for Poisson Distribution",
       sub=encode("Y(L) : [",?T(py)[1],?T(py)[11],"]     ",
       "X(R) : [",?T(px)[1],?T(px)[11],"]"))
    mtext(side=1,line=1,cex=2,outer=TRUE,
       encode("For Given X:",?T(x)," Y:",?T(y)," When Lambda =",
       ?T(r),"PMF =",?T(yy)))
    mtext(side=1,line=3,cex=2,outer=TRUE,
    "If you want to see the MLE for Lambda, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(rr)_seq(0,40,len=81)
    ?T(rr)_?T(rr)[?T(rr)>0]
    ?T(lx)_?mpoiss(rep(?T(x),len(?T(rr))),?T(rr))
    ?T(ly)_?mpoiss(rep(?T(y),len(?T(rr))),?T(rr))
    ?T(lr)_?T(lx)*?T(ly)
    ?T(er)_(?T(x)+?T(y))/2
    ?T(ml)_?mpoiss(?T(x),?T(er))*?mpoiss(?T(y),?T(er))
    par(mfrow=c(1,1), oma=c(0,3,0,0))
    plot(?T(rr),?T(lr),type="l",
       main="Maximum Likelihood for Poisson Distribution",
       sub=encode("For Given X:",?T(x)," Y:",?T(y)," When L =",
       ?T(er)," L(R) =",?T(ml)),xlab="Rambda",ylab="",err=-1)
    lines(c(?T(r),?T(r),0),c(0,?T(yy),?T(yy)))
    arrows(?T(er),?T(ml),0,?T(ml))
    arrows(?T(er),?T(ml),?T(er),0)
    mtext(side=2, line=2, outer=T, "MLE")
    rm(?T(x),?T(r),?T(y),?T(rr),?T(lr),?T(py),?T(px),
       ?T(z),?T(zx),?T(zy),?T(lx),?T(ly),?T(yy),?T(er),?T(ml))
 })
END
```

```
MACRO mlegeo
({
  item_c("Sample size 1",
         "Sample size 2",
         "Go to Previous Step")
  action_c("?mlegeo1",
           "?mlegeo2",
           "?mledisc")
  menu(item , action)
  rm(item, action)
})
END
```

```
MACRO mlegeo1(
x/?PROMPT(Random Sample ( 0 <= X <= 30 )         : )/,
p/?PROMPT(Estimated Probability ( 0 <= p <= 1 )  : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Geometric distribution.
#
({
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    ?T(x)_x
    ?T(p)_p
    ?T(y)_?mgeo(0:30,?T(p))
    ?T(iy)_?mgeo(?T(x),?T(p))
    plot(0:30,?T(y), main="PMF for Geometric Distribution",
       sub=encode("Given X :",?T(x)," When P =",?T(p)," PMF =",
       ?T(iy)), type="h", xlab="X-VALUE", ylab="",cex=.5,
       ylim=c(0,1),err=-1)
    points(?T(x),?T(iy),mark=0)
    mtext(side=2, line=2, outer=T, "PMF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for P, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp)_seq(0,1,len=101)
    ?T(yy)_?mgeo(?T(x),?T(pp))
    ?T(ep)_1/(?T(x)+1)
    ?T(ey)_?mgeo(?T(x),?T(ep))
    par(mfrow=c(1,1), oma=c(0,3,0,0))
    plot(?T(pp),?T(yy),type="l",xlab="P-value",ylab="",
       main="Maximum Likelihood for Geometric Distribution",
       sub=encode("Given X :",?T(x)," When P =",?T(ep),"L(P) =",
       ?T(ey)),cex=.5,err=-1)
    lines(c(?T(p),?T(p),0),c(0,?T(iy),?T(iy)))
    arrows(?T(ep),?T(ey),0,?T(ey))
    arrows(?T(ep),?T(ey),?T(ep),0)
    mtext(side=2, line=2, outer=T, "MLE")
    rm(?T(x),?T(p),?T(y),?T(iy),?T(pp),?T(yy),?T(ep),?T(ey))
 })
END
```

```
MACRO mlegeo2(
x/?PROMPT(Random Sample X ( 0 <= X <= 30 )          : )/,
y/?PROMPT(Random Sample Y ( 0 <= Y <= 30 )          : )/,
p/?PROMPT(Estimated Probability  ( 0 <= p <= 1)     : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Geometric diistribution(Sample size two).
#
({
    par(mfrow=c(1,1), oma=c(5,0,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(p)_p
    ?T(px)_max(0,?T(x)-5)
    ?T(px)_seq(?T(px),?T(px)+10)
    ?T(py)_max(0,?T(y)-5)
    ?T(py)_seq(?T(py),?T(py)+10)
    ?T(zx)_?mgeo(?T(px),?T(p))
    ?T(zx)_matrix(?T(zx),11,1)
    ?T(zy)_?mgeo(?T(py),?T(p))
    ?T(zy)_matrix(?T(zy),1,11)
    ?T(z)_?T(zx)%*?T(zy)
    ?T(z)_?T(z)/max(?T(z))
    ?T(iz)_?mgeo(?T(x),?T(p))*?mgeo(?T(y),?T(p))
    persp(?T(z))
    title(main="PMF for Geometric Distribution (Sample size Two)",
       sub=encode("Y(L) : [",?T(py)[1],?T(py)[11],"]     ",
       "X(R) : [",?T(px)[1],?T(px)[11],"]"))
    mtext(side=1,line=1,cex=2,outer=TRUE,
       encode("Given X :",?T(x),"Y :",?T(y)," When P =",?T(p),
       "PMF =",?T(iz)))
    mtext(side=1,line=3,cex=2,outer=TRUE,
    "If you want to see the MLE for P, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp)_seq(0,1,len=101)
    ?T(yy)_?mgeo(?T(x),?T(pp))*?mgeo(?T(y),?T(pp))
    ?T(ep)_2/(?T(x)+?T(y)+2)
    ?T(ey)_?mgeo(?T(x),?T(ep))*?mgeo(?T(y),?T(ep))
    par(mfrow=c(1,1), oma=c(0,3,0,0))
    plot(?T(pp),?T(yy),type="l",xlab="P-value",ylab="",
       main="Maximum Likelihood for Geometric Distribution",
       sub=encode("Given X:",?T(x),"Y:",?T(y)," When P =",?T(ep),
       "L(P) =",?T(ey)),cex=.5,err=-1)
    lines(c(?T(p),?T(p),0),c(0,?T(iz),?T(iz)))
    arrows(?T(ep),?T(ey),0,?T(ey))
    arrows(?T(ep),?T(ey),?T(ep),0)
    mtext(side=2, line=2, outer=T, "MLE")
    rm(?T(x),?T(y),?T(p),?T(px),?T(py),?T(z),?T(zx),?T(zy),
       ?T(iz),?T(pp),?T(yy),?T(ep),?T(ey))
 })
END
```

```
MACRO mlerey
({
  item_c("Sample size 1",
         "Sample size 2",
         "Go to Previous Step")
  action_c("?mlerey1",
           "?mlerey2",
           "?mledisc")
  menu( item , action)
  rm( item, action)
})
END



MACRO mlerey1
({
  item_c("1 Parameter for P1",
         "1 Parameter for P2",
         "2 Parameters",
         "Go to previous step")
  action_c("?mlerey1p1",
           "?mlerey1p2",
           "?mlerey1pb",
           "?mlerey1am")
  menu( item , action)
  rm( item, action)
})
END



MACRO mlerey2
({
  item_c("1 Parameter for P1",
         "1 Parameter for P2",
         "2 Parameters",
         "Go to previous step")
  action_c("?mlerey2p1",
           "?mlerey2p2",
           "?mlerey2pb",
           "?mlerey1am")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO mlerey1p1(
x/?PROMPT(Random Sample X (0,1 or 2)       :)/,
p2/?PROMPT(Given P2 ( 0 <= P2 <= 1 )        :)/,
p1/?PROMPT(Estimated P1 ( 0 <= P1 <= 1-P2 ) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for REyLmab distribution( For one sample with known P2).
#
({
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    ?T(x)_x
    ?T(p2)_p2
    ?T(p1)_p1
    if(?T(p1)+?T(p2)>1) fatal("P1 + P2 can't be greater than 1")
    ?T(xx)_c(0,1,2)
    ?T(y)_?mreylam(?T(xx),?T(p1),?T(p2))
    ?T(iy)_?mreylam(?T(x),?T(p1),?T(p2))
    ?T(my)_max(?T(y))
    plot(?T(xx),?T(y),type="h",ylim=c(0,1.1*?T(my)),
       main="PMF for ReyLamb Distribution",
       sub=encode("Given X:",?T(x)," P2:",?T(p2),
       "When P1 =",?T(p1),"PMF =",?T(iy)),
       xlim=c(-1,3),cex=.5,err=-1,xlab="X-Value",ylab="")
    points(?T(x),?T(iy),mark=0)
    arrows(-.2,?T(iy),-1,?T(iy))
    mtext(side=2, line=1, outer=T, "PMF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for P1, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp1)_seq(0,(1-?T(p2)),len=51)
    ?T(l)_?mreylam(?T(x),?T(pp1),?T(p2))
    ?T(ep)_ifelse(?T(x)!=2,(1-?T(p2)),0)
    ?T(ml)_?mreylam(?T(x),?T(ep),?T(p2))
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    plot(?T(pp1),?T(l),type="l",ylim=c(0,1.1*?T(ml)),xlim=c(-.2,1),
        main="Maximum Likelihood for ReyLamb Distribution",
        xlab="P1-value",ylab="",cex=.5,err=-1)
    mtext(side=2, line=1, outer=T, "MLE")
    if(?T(x)!=1) {
      lines(c(?T(p1),?T(p1),-.2),c(0,?T(iy),?T(iy)))
      arrows(?T(ep),?T(ml)/5,?T(ep),0)
      arrows(-.05,?T(ml),-.2,?T(ml))
      mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("For Given X:",?T(x)," P2:",?T(p2)," When P1 =",
      ?T(ep)," L(P1) =",?T(ml)))     }
    if(?T(x)==1) {
      mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("Given X:",?T(x)," P2:",?T(p2)," L(P1) =",
      ?T(p2)," for all possible P1"))     }
    rm(?T(x),?T(p1),?T(p2),?T(xx),?T(y),?T(iy),?T(pp1),?T(l),
       ?T(ml),?T(ep))
 })
END
```

```
MACRO mlerey1p2(
x/?PROMPT(Random Sample X (0,1 or 2)          :)/,
p1/?PROMPT(Given P1 ( 0 <= P1 <= 1 )          :)/,
p2/?PROMPT(Estimated P2 ( 0 <= P2 <= 1-P1 ) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for ReyLamb distribution( For one sample with known P1).
#
({
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    ?T(x)_x
    ?T(p1)_p1
    ?T(p2)_p2
    if(?T(p1)+?T(p2)>1) fatal("P1 + P2 can't be greater than 1")
    ?T(xx)_c(0,1,2)
    ?T(y)_?mreylam(?T(xx),?T(p1),?T(p2))
    ?T(iy)_?mreylam(?T(x),?T(p1),?T(p2))
    ?T(my)_max(?T(y))
    plot(?T(xx),?T(y),type="h",ylim=c(0,1.1*?T(my)),
       main="PMF for ReyLamb Distribution",
       sub=encode("Given X:",?T(x)," P1:",?T(p1),
       "When P2=",?T(p2),"PMF =",?T(iy)),
       xlim=c(-1,3),cex=.5,err=-1,xlab="X-Value",ylab="")
    points(?T(x),?T(iy),mark=0)
    arrows(-.2,?T(iy),-1,?T(iy))
    mtext(side=2, line=1, outer=T, "PMF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for P2, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp2)_seq(0,(1-?T(p1)),len=51)
    ?T(l)_?mreylam(?T(x),?T(p1),?T(pp2))
    ?T(ep)_ifelse(?T(x)!=2,(1-?T(p1)),0)
    ?T(ml)_?mreylam(?T(x),?T(p1),?T(ep))
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    plot(?T(pp2),?T(l),type="l",ylim=c(0,1.1*?T(ml)),xlim=c(-.2,1),
       main="Maximum Likelihood for ReyLamb Distribution",
       xlab="P2-value",ylab="",cex=.5,err=-1)
    mtext(side=2, line=1, outer=T, "MLE")
    if(?T(x)!=0) {
      lines(c(?T(p2),?T(p2),-.2),c(0,?T(iy),?T(iy)))
      arrows(?T(ep),?T(ml)/5,?T(ep),0)
      arrows(-.05,?T(ml),-.2,?T(ml))
      mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("Given X:",?T(x)," P1:",?T(p1)," When P2 =",
      ?T(ep)," L(P2) =",?T(ml)))     }
    if(?T(x)==0) {
      mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("Given X:",?T(x)," P1:",?T(p1)," L(P2) =",
      ?T(p1)," for all possible P2"))     }
    rm(?T(x),?T(p1),?T(p2),?T(xx),?T(y),?T(iy),?T(pp2),?T(l),
       ?T(ml),?T(ep))
 })
END
```

```
MACRO mlerey1pb(
x/?PROMPT(Random Sample X (0,1 or 2)       :)/,
p1/?PROMPT(Estimated P1 ( 0 <= P1 <= 1 )    :)/,
p2/?PROMPT(Estimated P2 ( 0 <= P2 <= 1-P1 ) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for ReyLamb distribution( For one sample with both unknown  P1 & P2).
#
({
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    ?T(x)_x
    ?T(p1)_p1
    ?T(p2)_p2
    if(?T(p1)+?T(p2)>1) fatal("P1 + P2 can't be greater than 1")
    ?T(xx)_c(0,1,2)
    ?T(y)_?mreylam(?T(xx),?T(p1),?T(p2))
    ?T(iy)_?mreylam(?T(x),?T(p1),?T(p2))
    ?T(my)_max(?T(y))
    plot(?T(xx),?T(y),type="h",ylim=c(0,1.1*?T(my)),
       main="PMF for ReyLamb Distribution",
       sub=encode("For Given X:",?T(x)," When P1:",?T(p1),
       " P2 =",?T(p2),"PMF =",?T(iy)),
       xlim=c(-1,3),cex=.5,err=-1,xlab="X-Value",ylab="")
    points(?T(x),?T(iy),mark=0)
    arrows(-.2,?T(iy),-1,?T(iy))
    mtext(side=2, line=1, outer=T, "PMF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for P1 and P2, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp1)_seq(0,1,len=11)
    ?T(pp2)_seq(0,1,len=11)
    ?T(ppo)_rep(?T(pp1),rep(11,11))
    ?T(ppt)_rep(?T(pp2),11)
    ?T(l)_?mreylam(?T(x),?T(ppo),?T(ppt))
    ?T(l)_matrix(?T(l),11,11,byrow=TRUE)
    ?T(l)[row(?T(l))+col(?T(l)) > 12]_0
    ?T(ml)_max(?T(l))
    ?T(mr)_?row(?T(l),max)
    ?T(mc)_?col(?T(l),max)
    ?T(r)_(1:len(?T(mr)))[?T(mr)==?T(ml)]
    ?T(c)_(1:len(?T(mc)))[?T(mc)==?T(ml)]
    ?T(e1)_?T(pp1)[?T(r)]
    ?T(e2)_?T(pp2)[?T(c)]
    ?T(l)_?T(l)/?T(ml)
    par(mfrow=c(1,1), oma=c(5,0,0,0))
    persp(?T(l))
    title(main="Maximum Likelihood for ReyLamb Distribution",
       sub="P2(left) : [ 0 1 ]  P1(right) : [ 0 1 ]")
    mtext(side=1,line=1,outer=TRUE,
       encode("Given X:",?T(x)," When P1 =",?T(e1)," P2 =",
       ?T(e2)," L(P1,P2) =",?T(ml)))
    rm(?T(x),?T(p1),?T(p2),?T(xx),?T(y),?T(iy),?T(my),?T(l), ?T(ml),
```

```
?T(pp1),?T(pp2),?T(ppo),?T(ppt),?T(mr),?T(mc),?T(r),?T(c),
?T(e1),?T(e2))
})
END
```

```
MACRO mlerey2p1(
x/?PROMPT(Random Sample X (0,1 or 2)        :)/,
y/?PROMPT(Random Sample Y (0,1 or 2)        :)/,
p2/?PROMPT(Given P2 ( 0 <= P2 <= 1 )         :)/,
p1/?PROMPT(Estimated P1 ( 0 <= P1 <= 1-P2 ) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for ReyLamb distribution( For two sample with known P2).
#
({
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(p2)_p2
    ?T(p1)_p1
    if(?T(p1)+?T(p2)>1) fatal("P1 + P2 can't be greater than 1")
    ?T(xx)_c(0,1,2)
    ?T(yy)_?T(xx)
    ?T(xx)_rep(?T(xx),rep(3,3))
    ?T(yy)_rep(?T(yy),3)
    ?T(z)_?mreylam(?T(xx),?T(p1),?T(p2))*?mreylam(?T(yy),?T(p1),?T(p2))
    ?T(iz)_?mreylam(?T(x),?T(p1),?T(p2))*?mreylam(?T(y),?T(p1),?T(p2))
    plot(?T(xx),?T(yy),type="n",xlim=c(-1,3),ylim=c(-1,3),
       err=-1,xlab="X-Value",ylab="")
    points(?T(x),?T(y),mark=1)
    ?T(k)_1
    for ( i in 0:2 ) {
      for( j in 0:2 ) {
        lines(c(i,i),c(j,j+?T(z)[?T(k)]))
        ?T(k)_?T(k)+1
        }
      }
    title(main="PMF for ReyLamb Distribution",
       sub=encode("Given X:",?T(x)," Y:",?T(y)," P2:",?T(p2),
       "When P1 =",?T(p1),"PMF =",?T(iz)))
    mtext(side=2, line=1, outer=T, "Y Values / PMF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for P1, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp)_seq(0,(1-?T(p2)),len=51)
    ?T(l)_?mreylam(?T(x),?T(pp),?T(p2))*?mreylam(?T(y),?T(pp),?T(p2))
    if( (?T(x)+?T(y)) < 2 ) ?T(ep)_1-?T(p2)
    if( (?T(x)+?T(y)) == 2 ) ?T(ep)_(1-?T(p2))/2
    if( (?T(x)+?T(y)) > 2 ) ?T(ep)_0
    ?T(ml)_?mreylam(?T(x),?T(ep),?T(p2))*?mreylam(?T(y),?T(ep),?T(p2))
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    plot(?T(pp),?T(l),type="l",ylim=c(0,1.1*?T(ml)),xlim=c(-.2,1),
       main="Maximum Likelihood for ReyLamb Distribution",
       xlab="P1-value",ylab="",cex=.5,err=-1)
    mtext(side=2, line=1, outer=T, "MLE")
    if(?T(x)!=1 | ?T(y)!=1) {
      lines(c(?T(p1),?T(p1),-.2),c(0,?T(iz),?T(iz)))
```

```
      arrows(?T(ep),?T(ml)/5,?T(ep),0)
      arrows(-.05,?T(ml),-.2,?T(ml))
      mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("For Given X:",?T(x)," Y:",?T(y)," P2:",?T(p2),
         " When P1 =", ?T(ep)," L(P1) =",?T(ml)))
      }
    if(?T(x)==1 & ?T(y)==1) {
      mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("Given X:",?T(x)," Y:",?T(y)," P2:",?T(p2)," L(P1) =",
         ?T(ml)," for all possible P1"))
      }
    rm(?T(x),?T(y),?T(p1),?T(p2),?T(xx),?T(yy),?T(z),?T(iz),?T(k),
       ?T(ml),?T(ep),?T(pp),?T(l))
 })
END
```

```
MACRO mlerey2p2(
x/?PROMPT(Random Sample X (0,1 or 2)        :)/,
y/?PROMPT(Random Sample Y (0,1 or 2)        :)/,
p1/?PROMPT(Given P1 ( 0 <= P1 <= 1 )         :)/,
p2/?PROMPT(Estimated P2 ( 0 <= P2 <= 1-P1 ) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for ReyLamb distribution( For two sample with known P1).
#
({
    par(mfrow=c(1,1), oma=c(5,1,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(p1)_p1
    ?T(p2)_p2
    if(?T(p1)+?T(p2)>1) fatal("P1 + P2 can't be greater than 1")
    ?T(xx)_c(0,1,2)
    ?T(yy)_?T(xx)
    ?T(xx)_rep(?T(xx),rep(3,3))
    ?T(yy)_rep(?T(yy),3)
    ?T(z)_?mreylam(?T(xx),?T(p1),?T(p2))*?mreylam(?T(yy),?T(p1),?T(p2))
    ?T(iz)_?mreylam(?T(x),?T(p1),?T(p2))*?mreylam(?T(y),?T(p1),?T(p2))
    plot(?T(xx),?T(yy),type="n",xlim=c(-1,3),ylim=c(-1,3),
       err=-1,xlab="X-Value",ylab="")
    points(?T(x),?T(y),mark=1)
    ?T(k)_1
    for ( i in 0:2 ) {
      for( j in 0:2 ) {
        lines(c(i,i),c(j,j+?T(z)[?T(k)]))
        ?T(k)_?T(k)+1
        }
      }
    title(main="PMF for ReyLamb Distribution",
       sub=encode("For Given X:",?T(x)," Y:",?T(y)," P1:",?T(p1),
       "When P2 =",?T(p2),"PMF =",?T(iz)))
    mtext(side=2, line=1, outer=T, "Y Value / PMF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for P2, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp)_seq(0,(1-?T(p1)),len=51)
    ?T(l)_?mreylam(?T(x),?T(p1),?T(pp))*?mreylam(?T(y),?T(p1),?T(pp))
    if( (?T(x)+?T(y))==2 | (?T(x)==1 & ?T(y)==1) ) ?T(ep)_1-?T(p1)
    if( (?T(x)+?T(y))==2 | (?T(x)+?T(y)==4) ) ?T(ep)_0
    if( (?T(x)+?T(y))==3 ) ?T(ep)_(1-?T(p1))/2
    ?T(ml)_?mreylam(?T(x),?T(p1),?T(ep))*?mreylam(?T(y),?T(p1),?T(ep))
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    plot(?T(pp),?T(l),type="l",ylim=c(0,1.1*?T(ml)),xlim=c(-.2,1),
       main="Maximum Likelihood for ReyLamb Distribution",
       xlab="P2-value",ylab="",cex=.5,err=-1)
    mtext(side=2, line=1, outer=T, "PMF")
    if(?T(x)!=0 | ?T(y)!=0) {
      lines(c(?T(p2),?T(p2),-.2),c(0,?T(iz),?T(iz)))
```

```
      arrows(?T(ep),?T(ml)/5,?T(ep),0)
      arrows(-.05,?T(ml),-.2,?T(ml))
      mtext(side=1,line=1,cex=2,outer=TRUE,
         encode("For Given X:",?T(x)," Y:",?T(y)," P1:",?T(p1),
         " When P2 =", ?T(ep)," L(P2) =",?T(ml)))
      }
    if(?T(x)==0 & ?T(y)==0) {
      mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("For Given X:",?T(x)," Y:",?T(y)," P1:",?T(p1)," L(P2) =",
         ?T(ml)," for all possible P2"))
      }
    rm(?T(x),?T(y),?T(p1),?T(p2),?T(xx),?T(yy),?T(z),?T(iz),?T(k),
       ?T(ml),?T(ep),?T(pp),?T(l))
 })
END
```

```
MACRO mlerey2pb(
x/?PROMPT(Random Sample X (0,1 or 2)       :)/,
y/?PROMPT(Random Sample Y (0,1 or 2)       :)/,
p1/?PROMPT(Estimated P1 ( 0 <= P1 <= 1 )    :)/,
p2/?PROMPT(Estimated P2 ( 0 <= P2 <= 1-P1 ) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for ReyLamb distribution( For two sample with both unknown P1 & P2).
#
({
    par(mfrow=c(1,1), oma=c(5,2,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(p1)_p1
    ?T(p2)_p2
    if(?T(p1)+?T(p2)>1) fatal("P1 + P2 can't be greater than 1")
    ?T(xx)_c(0,1,2)
    ?T(yy)_?T(xx)
    ?T(xx)_rep(?T(xx),rep(3,3))
    ?T(yy)_rep(?T(yy),3)
    ?T(z)_?mreylam(?T(xx),?T(p1),?T(p2))*?mreylam(?T(yy),?T(p1),?T(p2))
    ?T(iz)_?mreylam(?T(x),?T(p1),?T(p2))*?mreylam(?T(y),?T(p1),?T(p2))
    plot(?T(xx),?T(yy),type="n",xlim=c(-1,3),ylim=c(-1,3),
       err=-1,xlab="X-Value",ylab="")
    points(?T(x),?T(y),mark=1)
    ?T(k)_1
    for ( i in 0:2 ) {
      for( j in 0:2 ) {
        lines(c(i,i),c(j,j+?T(z)[?T(k)]))
        ?T(k)_?T(k)+1
        }
      }
    title(main="PMF for ReyLamb Distribution",
       sub=encode("For Given X:",?T(x)," Y:",?T(y)," When P1 =",
       ?T(p1), " P2 =",?T(p2),"PMF =",?T(iz)))
    mtext(side=2, line=1, outer=T, "Y Value / PMF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for P1 & P2, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(pp1)_seq(0,1,len=11)
    ?T(pp2)_seq(0,1,len=11)
    ?T(pp1)_rep(?T(pp1),rep(11,11))
    ?T(pp2)_rep(?T(pp2),11)
    ?T(l)_?mreylam(?T(x),?T(pp1),?T(pp2)) *
           ?mreylam(?T(y),?T(pp1),?T(pp2))
    ?T(l)_matrix(?T(l),11,11,byrow=TRUE)
    ?T(l)[row(?T(l))+col(?T(l)) > 12]_0
    if( (?T(x)+?T(y))==0 ) ?T(ep)_c(1,0)
    if( (?T(x)+?T(y))==1 ) ?T(ep)_c(.5,.5)
    if( (?T(x)==1) & (?T(y)==1) ) ?T(ep)_c(0,1)
    if( (?T(x)+?T(y))==2 & ?T(x)!=?T(y) ) ?T(ep)_c(.5,0)
    if( (?T(x)+?T(y))==3 ) ?T(ep)_c(0,.5)
```

```
    if( (?T(x)+?T(y))==4 ) ?T(ep)_c(0,0)
    ?T(e1)_?T(ep)[1]
    ?T(e2)_?T(ep)[2]
    ?T(ml)_?mreylam(?T(x),?T(e1),?T(e2))*?mreylam(?T(y),?T(e1),?T(e2))
    ?T(l)_?T(l)/max(?T(l))
    par(mfrow=c(1,1), oma=c(5,0,0,0))
    persp(?T(l))
    title(main="Maximum Likelihood for ReyLamb Distribution",
       sub="P2(left) : [ 0 1 ]  P1(right) : [ 0 1 ]")
    mtext(side=1,line=1,outer=TRUE,
       encode("For Given X:",?T(x)," Y:",?T(y)," When P1 =",?T(e1),
       " P2 =",?T(e2)," L(P1,P2) =",?T(ml)))
    rm(?T(x),?T(y),?T(p1),?T(p2),?T(xx),?T(yy),?T(iz),?T(z),?T(k),
       ?T(pp1),?T(pp2),?T(l),?T(ep),?T(e1),?T(e2),?T(ml))
 })
END
```

```
MACRO mledunif
({
  item_c("1 Parameter for Lower bound",
         "1 Parameter for Upper bound",
         "2 Parameters",
         "Go to previous step")
  action_c("?mledunifl",
           "?mledunifu",
           "?mledunifb",
           "?mledisc")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO mledunifl(x/?PROMPT(Random Sample X ( Any Integer ) :)/,
y/?PROMPT(Random Sample Y ( Any Integer ) :)/,
u/?PROMPT(Given Upper Bound                :)/,
l/?PROMPT(Estimated Lower Bound            :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Discrete Uniform distribution with known upper bound.
#
({
    par(mfrow=c(1,1), oma=c(7,0,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(rs)_c(?T(x),?T(y))
    ?T(u)_u
    ?T(l)_l
    ?T(bs)_max(?T(rs))
    ?T(ss)_min(?T(rs))
#
# If U < (X or Y), U can not be Upper bound for uniform distribution.
#
    if( ?T(u) < ?T(bs) )  {
       fatal("Given Upper Bound can't be Smaller than any one of
       given two samples")   }
    if( ?T(u) >= ?T(bs) )  {
    ?T(mx)_max(?T(bs),?T(u)) + 2
    ?T(mn)_min(?T(ss),?T(l)) - 2
    ?T(px)_seq(?T(mx),?T(mn))
    ?T(pmf)_1/(?T(u)-?T(l)+1)
# PMF for discrete uniform distribution is;
#     pmf = 1/(upper - lower +1)
    ?T(px)_ifelse( (?T(px)<=?T(u) & ?T(px)>=?T(l) ),?T(pmf),0)
    ?T(px)_matrix(?T(px),ncol=1)
    ?T(py)_t(?T(px))
    ?T(z)_?T(px)%*?T(py)
# Joint pmf for independent uniform distribution can be computed
# by the matrix form.
    ?T(pmf)_?T(pmf)^2
    ?T(z)_?T(z)/?T(pmf)
    persp(?T(z))
    ?T(pmf)_ifelse(?T(l)<=?T(ss) & ?T(u)>=?T(bs), ?T(pmf), 0)
    title(main="PMF for Discrete Uniform Distribution",
      sub=encode("Y(left) = X(right) : [",?T(mn),?T(mx),"]"))
    mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("Given Samples",?T(ss),?T(bs),"When Boundary = (",
      ?T(l),?T(u),")  PMF =",?T(pmf)))
    mtext(side=1,line=3,cex=2,outer=TRUE,
    "If you want to see the MLE for Lower Bound, hit Return
    when GO? appears.  If not, hit the Break key")
    ?T(b)_seq(?T(ss),(?T(ss)-8))
# Select the range of lower bound
    ?T(sb)_min(?T(b))
    ?T(bb)_max(?T(b))
```

```
    ?T(pmf)_(1/(abs(?T(u)-?T(b))+1))^2
    ?T(ml)_max(?T(pmf))
# MLE can be determined when maximum pmf is found.
    ?T(mb)_?T(b)[(1:len(?T(pmf)))[?T(pmf)==?T(ml)]]
# MLE is located in the same position in the vector as the max pmf.
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    plot(?T(b),?T(pmf),type="h",ylim=c(0,1.1*?T(ml)),
    xlim=c(?T(sb)-1,?T(bb)+1),xlab="Parameter",ylab="",
       main="Maximum Likelihood for Discrete Uniform Distribution",
       sub=encode("Given Samples",?T(ss),?T(bs)," Upper Bound :",
       ?T(u)),cex=.5,err=-1)
    mtext(side=2, line=2, outer=T, "PMF")
    mtext(side=1,line=1,outer=TRUE,
       encode("When Lower Bound =",?T(mb)," L(P) =",?T(ml)))
    arrows(?T(sb)+(?T(sb)-?T(sb))/10,?T(ml),?T(sb)-1,?T(ml))
    rm(?T(x),?T(y),?T(rs),?T(l),?T(u),?T(bs),?T(ss),?T(mx),?T(b),
       ?T(mn),?T(px),?T(pmf),?T(py),?T(z),?T(sb),?T(bb),?T(ml),?T(mb))
    }
})
END
```

```
MACRO mledunifu(
x/?PROMPT(Random Sample X ( Any Integer ) :)/,
y/?PROMPT(Random Sample Y ( Any Integer ) :)/,
l/?PROMPT(Given Lower Bound              :)/,
u/?PROMPT(Estimated Upper Bound          :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Discrete Uniform distribution with known lower bound.
#
{{
    par(mfrow=c(1,1), oma=c(7,0,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(rs)_c(?T(x),?T(y))
    ?T(l)_l
    ?T(u)_u
    ?T(bs)_max(?T(rs))
    ?T(ss)_min(?T(rs))
#
# If L > (X or Y), L can not be lower bound for uniform distribution.
#
     if( ?T(l) > ?T(ss) )  {
        fatal("Given Lower Bound can't be Greater than any one of
        given two samples")   }
     if( ?T(l) <= ?T(ss) )  {
    ?T(mx)_max(?T(bs),?T(u)) + 2
    ?T(mn)_min(?T(ss),?T(l)) - 2
    ?T(px)_seq(?T(mx),?T(mn))
    ?T(pmf)_1/(?T(u)-?T(l)+1)
# PMF for discrete uniform distribution is;
#     pmf = 1/(upper - lower +1)
    ?T(px)_ifelse( (?T(px)<=?T(u) & ?T(px)>=?T(l) ),?T(pmf),0)
    ?T(px)_matrix(?T(px),ncol=1)
    ?T(py)_t(?T(px))
    ?T(z)_?T(px)%*?T(py)
# Joint pmf for independent uniform distribution can be computed
# by the matrix form.
    ?T(pmf)_?T(pmf)^2
    ?T(z)_?T(z)/?T(pmf)
    persp(?T(z))
    ?T(pmf)_ifelse(?T(l)<=?T(ss) & ?T(u)>=?T(bs), ?T(pmf), 0)
    title(main="PMF for Discrete Uniform Distribution",
       sub=encode("Y(left) = X(right) : [",?T(mn),?T(mx),"]"))
    mtext(side=1,line=1,cex=2,outer=TRUE,
       encode("Given Samples",?T(ss),?T(bs),"When Boundary = (",
       ?T(l),?T(u),")  PMF =",?T(pmf)))
    mtext(side=1,line=3,cex=2,outer=TRUE,
    "If you want to see the MLE for Upper Bound, hit Return
    when GO? appears.  If not, hit the Break key")
    ?T(b)_seq(?T(bs),(?T(bs)+8))
# Select the range of Upper Bound
    ?T(sb)_min(?T(b))
```

```
   ?T(bb)_max(?T(b))
   ?T(pmf)_(1/(abs(?T(l)-?T(b))+1))^2
   ?T(ml)_max(?T(pmf))
# MLE can be determined when maximum pmf is found.
   ?T(mb)_?T(b)[(1:len(?T(pmf)))[?T(pmf)==?T(ml)]]
# MLE is located in the same position in the vector as the max pmf.
   par(mfrow=c(1,1), oma=c(5,3,0,0))
   plot(?T(b),?T(pmf),type="h",ylim=c(0,1.1*?T(ml)),
      xlim=c(?T(sb)-1,?T(bb)+1),xlab="Parameter",ylab="",
      main="Maximum Likelihood for Discrete Uniform Distribution",
      sub=encode("For Given Samples",?T(ss),?T(bs),"  Lower Bound :",
      ?T(l)),cex=.5,err=-1)
   mtext(side=2, line=1, outer=T, "PMF")
   mtext(side=1,line=1,outer=TRUE,
      encode("When Upper Bound =",?T(mb)," L(P) =",?T(ml)))
   arrows(?T(sb)+(?T(sb)-?T(sb))/10,?T(ml),?T(sb)-1,?T(ml))
   rm(?T(x),?T(y),?T(rs),?T(l),?T(u),?T(bs),?T(ss),?T(mx),?T(b),
      ?T(mn),?T(px),?T(pmf),?T(py),?T(z),?T(sb),?T(bb),?T(ml),?T(mb))
   }
})
END
```

```
MACRO mledunifb(
x/?PROMPT(Random Sample X ( Any Integer ) :)/,
y/?PROMPT(Random Sample Y ( Any Integer ) :)/,
l/?PROMPT(Estimated Lower Bound          :)/,
u/?PROMPT(Estimated Upper Bound          :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Discrete Uniform distribution with unknown parameter.
#
({
    par(mfrow=c(1,1), oma=c(7,0,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(rs)_c(?T(x),?T(y))
    ?T(l)_l
    ?T(u)_u
    ?T(bs)_max(?T(rs))
    ?T(ss)_min(?T(rs))
    ?T(mx)_max(?T(bs),?T(u)) + 2
    ?T(mn)_min(?T(ss),?T(l)) - 2
    ?T(px)_seq(?T(mx),?T(mn))
    ?T(pmf)_1/(?T(u)-?T(l)+1)
# PMF for discrete uniform distribution is;
#     pmf = 1/(upper - lower +1)
    ?T(px)_ifelse( (?T(px)<=?T(u) & ?T(px)>=?T(l) ),?T(pmf),0)
    ?T(px)_matrix(?T(px),ncol=1)
    ?T(py)_t(?T(px))
    ?T(z)_?T(px)%*?T(py)
# Joint pmf for independent uniform distribution can be computed
# by the matrix form.
    ?T(pmf)_?T(pmf)^2
    ?T(z)_?T(z)/?T(pmf)
    persp(?T(z))
    ?T(pmf)_ifelse(?T(l)<=?T(ss) & ?T(u)>=?T(bs), ?T(pmf), 0)
    title(main="PMF for Discrete Uniform Distribution",
      sub=encode("Y(left) = X(right) : [",?T(mn),?T(mx),"]"))
    mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("Given Samples",?T(ss),?T(bs),"When Boundary = (",
      ?T(l),?T(u),")  PMF =",?T(pmf)))
    mtext(side=1,line=3,cex=2,outer=TRUE,
    "If you want to see the MLE for Upper/Lower Bound,
    hit Return when GO? appears.  If not, hit the Break key")
    ?T(u)_seq(?T(bs),?T(bs)+5)
    ?T(l)_seq(?T(ss)-5,?T(ss))
# Select the range of Upper and Lower Bound
    ?T(u)_rep(?T(u),rep(6,6))
    ?T(l)_rep(?T(l),6)
    ?T(pmf)_(1/(?T(u)-?T(l)+1))^2
    ?T(ml)_max(?T(pmf))
    ?T(zz)_matrix(?T(pmf),6,6,byrow=TRUE)
    ?T(zz)_?T(zz)/?T(ml)
    par(mfrow=c(1,1), oma=c(5,0,0,0))
```

```
    persp(?T(zz))
    title(main="Maximum Likelihood for Discrete Uniform Distribution",
      encode("Lower Bound(L) : [",?T(ss)-5,?T(ss),"]  ",
      "Upper Bound(R) : [",?T(bs),?T(bs)+5,"]"))
    mtext(side=1,line=1,outer=TRUE,
      encode("Given Samples",?T(ss),?T(bs),
      "When Boundary = (",?T(ss),?T(bs),")  L(P) =",?T(ml)))
    rm(?T(x),?T(y),?T(rs),?T(l),?T(u),?T(bs),?T(ss),?T(mx),
       ?T(mn),?T(px),?T(pmf),?T(py),?T(z),?T(ml),?T(zz))
})
END
```

```
MACRO mlecont
({
  item_c("Normal       Distribution",
         "Lognormal    Distribution",
         "Exponential  Distribution",
         "Uniform      Distribution",
         "Standard Gamma Distribution",
         "Go to previous step")
  action_c("?mlenorm",
           "?mlelnorm",
           "?mleexpon",
           "?mlecunif",
           "?mlestgam",
           "?mle")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO mlenorm
({
  item_c("Sample size 1",
         "Sample size 2",
         "Go to Previous Step")
  action_c("?mlenorm1",
           "?mlenorm2",
           "?mlecont")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO mlenorm1
({
  item_c("1 Parameter for Mean",
         ". Parameter for Sigma",
         "2 Parameters",
         "Go to previous step")
  action_c("?mlenorm1m",
           "?mlenorm1s",
           "?mlenorm1b",
           "?mlenorm")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO mlenorm2
({
  item_c("1 Parameter for Mean",
         "1 Parameter for Sigma",
         "2 Parameters",
         "Go to previous step")
  action_c("?mlenorm2m",
           "?mlenorm2s",
           "?mlenorm2b",
           "?mlenorm")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO mlenorm1m(
x/?PROMPT(Random Sample X        :)/,
s/?PROMPT(Given Sigma ( S > 0 ) :)/,
m/?PROMPT(Estimated Mean Value   :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Normal distribution( For one sample with known Sigma).
#
({
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    ?T(x)_x
    ?T(s)_s
    ?T(m)_m
    ?T(es)_abs(?T(x)-?T(m))+?T(s)
    ?T(xx)_seq(?T(m)-?T(es),?T(m)+?T(es),len=51)
    ?T(xl)_floor(min(?T(xx)))
    ?T(xr)_ceiling(max(?T(xx)))
    ?T(y)_dnorm(?T(xx),?T(m),?T(s))
    ?T(my)_dnorm(?T(m),?T(m),?T(s))
    ?T(iy)_dnorm(?T(x),?T(m),?T(s))
    plot(?T(xx),?T(y),type="l",xlab="X-Value",ylab="",
      main=encode("PDF for Given Sigma :",?T(s)," Estimated Mu :",
      ?T(m)), sub=encode("When X =",?T(x)," PDF =",?T(iy)),
      xlim=c(?T(xl),?T(xr)),ylim=c(0,1.1*?T(my)),cex=.5,err=-1)
    lines(c(?T(x),?T(x),?T(xl)),c(0,?T(iy),?T(iy)))
    arrows(?T(m),?T(my),?T(m),0)
    arrows(?T(m),?T(my),?T(xl),?T(my))
    mtext(side=2, line=1, outer=T, "PDF")
    mtext(side=1,line=1,outer=TRUE,
    "If you want to see the MLE for Mean,
    hit Return when GO? appears.  If not, hit the Break key")
    ?T(mm)_seq(?T(x)-1.5*?T(es),?T(x)+1.5*?T(es),len=51)
    ?T(nm)_len(?T(mm))
    ?T(l)_dnorm(rep(?T(x),?T(nm)),?T(mm),rep(?T(s),?T(nm)))
    ?T(xl)_floor(min(?T(mm)))
    ?T(xr)_ceiling(max(?T(mm)))
    ?T(ml)_dnorm(?T(x),?T(x),?T(s))
    par(mfrow=c(1,1), oma=c(0,3,0,0))
    plot(?T(mm),?T(l),type="l",xlim=c(?T(xl),?T(xr)),
          xlab="Mu",ylab="",ylim=c(0,1.1*?T(ml)),
          main="Maximum Likelihood Estimate for Normal Distribution",
          sub=encode("Given X :",?T(x),"S :",?T(s),"When Mu =",
          ?T(x),"L(M) =",?T(ml)),cex=.5,err=-1)
    mtext(side=2, line=2, outer=T, "MLE")
    lines(c(?T(m),?T(m),?T(xl)),c(0,?T(iy),?T(iy)))
    arrows(?T(x),?T(ml),?T(x),0)
    arrows(?T(x),?T(ml),?T(xl),?T(ml))
    rm(?T(x),?T(s),?T(m),?T(nm),?T(y),?T(xl),?T(xr),?T(es),?T(xx),
       ?T(my),?T(iy),?T(mm),?T(l),?T(ml))
 })
END
```

```
MACRO mlenorm1s(
x/?PROMPT(Random Sample X       :)/,
m/?PROMPT(Given Mu              :)/,
s/?PROMPT(Estimated Sigma (S>0) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Normal distribution( For one sample with known Mu).
#
({
    ?T(x)_x
    ?T(m)_m
    ?T(s)_s
    if(?T(x)==?T(m)) {
    print ("When X and Mu are same,")
    print ("the only possible Sigma is zero")
    }
    if(?T(x)!=?T(m)) {
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    ?T(es)_abs(?T(x)-?T(m))
    ?T(xx)_seq(?T(x)-1.5*?T(es),?T(x)+1.5*?T(es),len=51)
    ?T(iy)_dnorm(?T(x),?T(m),?T(s))
    ?T(im)_dnorm(?T(m),?T(m),?T(s))
    ?T(l)_dnorm(?T(xx),?T(m),?T(s))
    ?T(xl)_floor(min(?T(xx)))
    ?T(xr)_ceiling(max(?T(xx)))
    plot(?T(xx),?T(l),type="l",xlim=c(?T(xl),?T(xr)),
          xlab="X",ylab="" ,ylim=c(0,1.1*?T(im)),
          main=encode("PDF for Given Mu :",?T(m),"Estimated S :",?T(s)),
          sub=encode("When X is",?T(x),"  PDF is",?T(iy)),cex=.5,err=-1)
    lines(c(?T(x),?T(x),?T(xl)),c(0,?T(iy),?T(iy)))
    arrows(?T(m),?T(im),?T(m),0)
    arrows(?T(m),?T(im),?T(xl),?T(im))
    mtext(side=2, line=1, outer=T, "PDF")
    mtext(side=1,line=1,outer=TRUE,
    "If you want to see the MLE for Sigma, range S+/-2,
    hit Return when GO? appears.  If not, hit the Break key")
    ?T(sl)_max(0,?T(s)-2)
    ?T(sr)_?T(s)+2
    ?T(ss)_seq(?T(sl),?T(sr),len=101)
    ?T(ss)_?T(ss)[?T(ss)>0]
    ?T(ns)_len(?T(ss))
    ?T(y)_dnorm(rep(?T(x),?T(ns)),rep(?T(m),?T(ns)),?T(ss))
    ?T(xl)_floor(?T(sl))
    ?T(xr)_ceiling(?T(sr))
    ?T(ml)_dnorm(?T(x),?T(m),?T(es))
     if ( ?T(es) >= ?T(sl) & ?T(es) <= ?T(sr) )  {
    par(mfrow=c(1,1), oma=c(0,3,0,0))
    plot(?T(ss),?T(y),type="l",xlim=c(?T(xl),?T(xr)),
          xlab="Sigma",ylab="",ylim=c(0,1.1*?T(ml)),
          main="Maximum Likelihood Estimate for Normal Distribution",
          sub=encode("Given X :",?T(x),"Mu :",?T(m),"When S =",?T(es),
          "L(S) =",?T(ml)),cex=.5,err=-1)
```

```
   mtext(side=2, line=1, outer=T, "MLE")
   lines(c(?T(s),?T(s),?T(xl)),c(0,?T(iy),?T(iy)))
   arrows(?T(es),?T(ml),?T(es),0)
   arrows(?T(es),?T(ml),?T(xl),?T(ml))
        }
   if ( ?T(es) < ?T(sl) | ?T(es) > ?T(sr) )  {
   par(mfrow=c(1,1), oma=c(5,3,0,0))
   plot(?T(ss),?T(y),type="l",xlim=c(?T(xl),?T(xr)),
         xlab="Sigma",ylab="",ylim=c(0,max(?T(y))),
         main="Maximum Likelihood Estimate for Normal Distribution",
         sub="There is no Sigma which will make MLE",)
   lines(c(?T(s),?T(s),?T(xl)),c(0,?T(iy),?T(iy)))
   mtext(side=2, line=1, outer=T, "MLE")
   mtext(side=1,line=1,outer=TRUE,
   "If you want to see the MLE, hit Return when GO? appears
    If not, hit the Break key")
   ?T(ss)_seq(0,1.5*?T(es),len=51)
   ?T(ss)_?T(ss)[?T(ss)>0]
   ?T(xr)_ceiling(max(?T(ss)))
   ?T(ns)_len(?T(ss))
   ?T(y)_dnorm(rep(?T(x),?T(ns)),rep(?T(m),?T(ns)),?T(ss))
   par(mfrow=c(1,1), oma=c(0,3,0,0))
   plot(?T(ss),?T(y),type="l",
         xlab="Sigma",ylab="",ylim=c(0,1.1*?T(ml)),
         main="Maximum Likelihood Estimate for Normal Distribution",
         sub=encode("Given X :",?T(x),"Mu :",?T(m),"  When S =",
         ?T(es),"L(S) =",?T(ml)),cex=.5,err=-1)
   mtext(side=2, line=1, outer=T, "MLE")
   lines(c(?T(s),?T(s),0),c(0,?T(iy),?T(iy)))
   arrows(?T(es),?T(ml),?T(es),0)
   arrows(?T(es),?T(ml),0,?T(ml))
   }
   rm(?T(sl),?T(sr),?T(ss),?T(ns),?T(y),?T(im),
      ?T(xl),?T(xr),?T(es),?T(ml),?T(xx),?T(iy))
  }
 rm(?T(x),?T(m),?T(s))
 })
END
```

```
MACRO mlenorm1b(x/?PROMPT(Random Sample X :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Normal distribution( For one sample with unknown Mu & Sigma).
#
({
    par(mfrow=c(1,1), oma=c(5,0,0,0))
    ?T(x)_x
    ?T(s)_1:11
    ?T(m)_seq(?T(x)-5,?T(x)+5,len=11)
    ?T(m)_rep(?T(m),rep(11,11))
    ?T(s)_rep(?T(s),11)
    ?T(z)_dnorm(rep(?T(x),121),?T(m),?T(s))
    ?T(pz)_?T(z)/max(?T(z))
    ?T(pz)_matrix(?T(pz),11,11,byrow=TRUE)
    persp(?T(pz))
    title(main="Maximum Likelihood Estimate for Normal
      Distribution",sub=encode("Given X is",?T(x),
      "Sigma (Left) 1 : 11,  Mu (Right)",?T(x)-5,":",?T(x)+5))
    mtext(side=1,line=1,outer=TRUE,
      encode("Assume min Sigma = 1, when Mu =",?T(x),"S = 1   L(M,S) =",
      max(?T(z))))
    rm(?T(x),?T(s),?T(m),?T(z),?T(pz))
 })
END
```

```
MACRO mlenorm2m(
x/?PROMPT(Random Sample X          : )/,
y/?PROMPT(Random Sample Y          : )/,
r/?PROMPT(Correlation ( |R| < 1 )  : )/,
s/?PROMPT(Given Sigma ( S > 0 )    : )/,
m/?PROMPT(Estimated Mu             : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Normal distribution( For two sample with known Sigma).
#
({
    par(mfrow=c(1,1), oma=c(5,0,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(r)_r
    ?T(s)_s
    ?T(m)_m
    ?T(dx)_(?T(x)-5):(?T(x)+5)
    ?T(dy)_(?T(y)-5):(?T(y)+5)
    ?T(dx)_rep(?T(dx),rep(11,11))
    ?T(dy)_rep(?T(dy),11)
    ?T(z)_?dbinorm(?T(m),?T(s),?T(m),?T(s),?T(r),?T(dx),?T(dy))
    ?T(pz)_?T(z)/max(?T(z))
    ?T(pz)_matrix(?T(pz),11,11,byrow=TRUE)
    ?T(iz)_?dbinorm(?T(m),?T(s),?T(m),?T(s),?T(r),?T(x),?T(y))
    persp(?T(pz))
    title(main=encode("PDF Surface for Given S:",?T(s),"R:",?T(r),
      "Estimated Mu:",?T(m)),
    sub=encode("When X=",?T(x),"(R/center) Y=",?T(y),"(L/center)",
      " PDF =",?T(iz)))
    mtext(side=1,line=1,outer=TRUE,
    "If you want to see the MLE for Mu, hit Return
    when GO? appears.  If not, hit the Break key")
    ?T(mt)_(?T(x)+?T(y))/2
    ?T(df)_abs(?T(mt)-?T(m))+?T(s)
    ?T(mm)_seq(?T(mt)-?T(df),?T(mt)+?T(df),len=51)
    ?T(l)_?dbinorm(?T(mm),?T(s),?T(mm),?T(s),?T(r),?T(x),?T(y))
    ?T(xll)_floor(min(?T(mm)))
    ?T(xlr)_ceiling(max(?T(mm)))
    ?T(yl)_max(?T(l))
    ?T(ml)_?dbinorm(?T(mt),?T(s),?T(mt),?T(s),?T(r),?T(x),?T(y))
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    plot(?T(mm),?T(l),type="l",xlim=c(?T(xll),?T(xlr)),
          xlab="Mu",ylab="",ylim=c(0,1.1*?T(yl)),
          main="Maximum Likelihood Estimate for Normal Distribution",
          sub=encode("Given X :",?T(x),"Y :",?T(y),"S :",?T(s),"R :",
          ?T(r)),cex=.5,err=-1)
    mtext(side=2, line=1, outer=T, "MLE")
    lines(c(?T(m),?T(m),?T(xll)),c(0,?T(iz),?T(iz)))
    arrows(?T(mt),?T(yl),?T(mt),0)
    arrows(?T(mt),?T(yl),?T(xll),?T(yl))
    mtext(side=1,line=1,cex=2,outer=TRUE,
```

```
    encode("When MU =",?T(mt),"  L(M) =",?T(ml)))
    rm(?T(x),?T(y),?T(r),?T(s),?T(m),?T(xll),?T(xlr),?T(yl),?T(df),
       ?T(dx),?T(dy),?T(z),?T(pz),?T(mt),?T(mm),?T(l),?T(ml))
})
END
```

```
MACRO mlenorm2s(
x/?PROMPT(Random Sample X          : )/,
y/?PROMPT(Random Sample Y          : )/,
r/?PROMPT(Correlation ( |R| < 1 )  : )/,
m/?PROMPT(Given Mu                 : )/,
s/?PROMPT(Estimated Sigma ( S>0 )  : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Normal distribution( For two sample with known Mu).
#
({
    ?T(x)_x
    ?T(y)_y
    ?T(r)_r
    ?T(m)_m
    ?T(s)_s
    if( (?T(x)==?T(m)) & (?T(y)==?T(m)) ) {
    print ("When X and Y and Mu are all same,")
    print ("Sigma = zero is the MLE")
    }
    if( (?T(x)!=?T(m)) | (?T(y)!=?T(m)) ) {
    par(mfrow=c(1,1), oma=c(7,0,0,0))
    ?T(xx)_seq(?T(x)-2*?T(s),?T(x)+2*?T(s),len=11)
    ?T(sx)_?T(xx)[1]
    ?T(bx)_?T(xx)[11]
    ?T(dx)_?T(xx)[2]-?T(xx)[1]
    ?T(yy)_seq(?T(y)-2*?T(s),?T(y)+2*?T(s),len=11)
    ?T(sy)_?T(yy)[1]
    ?T(by)_?T(yy)[11]
    ?T(dy)_?T(yy)[2]-?T(yy)[1]
    ?T(xx)_rep(?T(xx),rep(11,11))
    ?T(yy)_rep(?T(yy),11)
    ?T(z)_?dbinorm(?T(m),?T(s),?T(m),?T(s),?T(r),?T(xx),?T(yy))
    ?T(iz)_?dbinorm(?T(m),?T(s),?T(m),?T(s),?T(r),?T(x),?T(y))
    ?T(z)_?T(z)/max(?T(z))
    ?T(z)_matrix(?T(z),11,11,byrow=TRUE)
    persp(?T(z))
    title(main=encode("PDF for Given Mu :",?T(m),"Estimated S :",?T(s)),
       sub=encode("Y(L) [",?T(sy),?T(by),"] Dy =",?T(dy),
       "X(R) [",?T(sx),?T(bx),"] Dx =",?T(dx)))
    mtext(side=1,line=1,outer=TRUE,
       encode("When X is",?T(x)," Y is",?T(y),"PDF is",?T(iz)))
    mtext(side=1,line=3,outer=TRUE,
    "If you want to see the MLE for Sigma, range S+/-2,
    hit Return when GO? appears.  If not, hit the Break key")
    ?T(sl)_max(0,?T(s)-2)
    ?T(sr)_?T(s)+2
    ?T(ss)_seq(?T(sl),?T(sr),len=101)
    ?T(ss)_?T(ss)[?T(ss)>0]
    ?T(zz)_?dbinorm(?T(m),?T(ss),?T(m),?T(ss),?T(r),?T(x),?T(y))
    ?T(xll)_floor(?T(sl))
    ?T(xlr)_ceiling(?T(sr))
```

```
?T(xm)_?T(x)-?T(m)
?T(ym)_?T(y)-?T(m)
?T(es)_sqrt((?T(xm)^2-2*?T(r)*?T(xm)*?T(ym)+?T(ym)^2) /
           (2*(1-?T(r)^2)))
?T(ml)_?dbinorm(?T(m),?T(es),?T(m),?T(es),?T(r),?T(x),?T(y))
par(mfrow=c(1,1), oma=c(5,3,0,0))
plot(?T(ss),?T(zz),type="l",xlim=c(?T(xll),?T(xlr)),
      xlab="Sigma",ylab="")
mtext(side=2, line=1, outer=T, "MLE")
lines(c(?T(s),?T(s),?T(xll)),c(0,?T(iz),?T(iz)))
 if ( ?T(es) >= ?T(sl) & ?T(es) <= ?T(sr) )  {
 title(main="Maximum Likelihood Estimate for Normal Distribution",
      sub=encode("Given X:",?T(x),"Y:",?T(y),"Mu:",?T(m),"When S=",
      ?T(es),"L(S)=",?T(ml)),cex=.5,err=-1)
 arrows(?T(es),?T(ml)/5,?T(es),0)
 arrows(?T(xll)+(?T(xlr)-?T(xll))/5,?T(ml),?T(xll),?T(ml))
     }
 if ( ?T(es) < ?T(sl) | ?T(es) > ?T(sr) )  {
 title(main="Maximum Likelihood Estimate for Normal Distribution",
      sub="There is no Sigma which will make MLE",)
mtext(side=1,line=1,outer=TRUE,
"If you want to see the MLE, hit Return when GO? appears
 If not, hit the Break key")
?T(sm)_seq(0,1.5*?T(es),len=51)
?T(sm)_?T(sm)[?T(sm)>0]
?T(xrr)_ceiling(max(?T(sm)))
?T(zt)_?dbinorm(?T(m),?T(sm),?T(m),?T(sm),?T(r),?T(x),?T(y))
par(mfrow=c(1,1), oma=c(0,3,0,0))
plot(?T(sm),?T(zt),type="l",
      xlab="Sigma",ylab="",ylim=c(0,1.1*?T(ml)),
      main="Maximum Likelihood Estimate for Normal Distribution",
      sub=encode("Given X:",?T(x),"Y:",?T(y),"Mu:",?T(m),
      "When S=",?T(es),"L(S)=",?T(ml)),cex=.5,err=-1)
mtext(side=2, line=1, outer=T, "MLE")
lines(c(?T(s),?T(s),?T(xll)),c(0,?T(iz),?T(iz)))
 arrows(?T(es),?T(ml)/5,?T(es),0)
 arrows(?T(xrr)/5,?T(ml),0,?T(ml))
rm(?T(sm),?T(xrr),?T(zt))
 }
 rm(?T(sl),?T(sr),?T(ss),?T(xx),?T(yy),?T(z),?T(iz),?T(xm),?T(ym),
    ?T(xll),?T(xlr),?T(es),?T(ml),?T(zz))
    }
rm(?T(x),?T(y),?T(m),?T(s),?T(r),?T(sx),?T(bx),?T(dx),
   ?T(sy),?T(by),?T(dy))
})
END
```

```
MACRO mlenorm2b(
x/?PROMPT(Random Sample X          : )/,
y/?PROMPT(Random Sample Y          : )/,
r/?PROMPT(Correlation ( |R| < 1 ) : )/,
m/?PROMPT(Estimated Mu             : )/,
s/?PROMPT(Estimated Sigma ( S>0 ) : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Normal distribution( For two sample with known Mu).
#
({
    ?T(x)_x
    ?T(y)_y
    ?T(r)_r
    ?T(m)_m
    ?T(s)_s
    par(mfrow=c(1,1), oma=c(7,0,0,0))
    ?T(xx)_seq(?T(x)-2*?T(s),?T(x)+2*?T(s),len=11)
    ?T(sx)_?T(xx)[1]
    ?T(bx)_?T(xx)[11]
    ?T(dx)_?T(xx)[2]-?T(xx)[1]
    ?T(yy)_seq(?T(y)-2*?T(s),?T(y)+2*?T(s),len=11)
    ?T(sy)_?T(yy)[1]
    ?T(by)_?T(yy)[11]
    ?T(dy)_?T(yy)[2]-?T(yy)[1]
    ?T(xx)_rep(?T(xx),rep(11,11))
    ?T(yy)_rep(?T(yy),11)
    ?T(z)_?dbinorm(?T(m),?T(s),?T(m),?T(s),?T(r),?T(xx),?T(yy))
    ?T(iz)_?dbinorm(?T(m),?T(s),?T(m),?T(s),?T(r),?T(x),?T(y))
    ?T(z)_?T(z)/max(?T(z))
    ?T(z)_matrix(?T(z),11,11,byrow=TRUE)
    persp(?T(z))
    title(main=encode("PDF for Estimated Mu :",?T(m)," S :",?T(s)),
       sub=encode("Y(L) [",?T(sy),?T(by),"] Dy =",?T(dy),
       "X(R) [",?T(sx),?T(bx),"] Dx =",?T(dx)))
    mtext(side=1,line=1,outer=TRUE,
    encode("Given X :",?T(x)," Y :",?T(y),"PDF =",?T(iz)))
    mtext(side=1,line=3,outer=TRUE,
    "If you want to see the MLE for Mu and Sigma,
    hit Return when GO? appears.  If not, hit the Break key")
    if( ?T(x)==?T(y) ) {
    print ("When X and Y are same,")
    print ("Mu = X(or Y) and Sigma = zero is the MLE")
    }
    if( ?T(x)!=?T(y) ) {
    ?T(em)_(?T(x)+?T(y))/2
    ?T(xm)_?T(x)-?T(em)
    ?T(ym)_?T(y)-?T(em)
    ?T(es)_sqrt((?T(xm)^2-2*?T(r)*?T(xm)*?T(ym)+?T(ym)^2) /
              (2*(1-?T(r)^2)))
    ?T(mm)_seq(?T(m)-?T(s),?T(m)+?T(s),len=11)
    ?T(lm)_?T(mm)[1]
```

```
?T(bm)_?T(mm)[11]
?T(dm)_?T(mm)[2]-?T(mm)[1]
?T(ss)_seq(.5*?T(s),1.5*?T(s),len=11)
?T(ls)_?T(ss)[1]
?T(bs)_?T(ss)[11]
?T(ds)_?T(ss)[2]-?T(ss)[1]
?T(mz)_?dbinorm(?T(em),?T(es),?T(em),?T(es),?T(r),?T(x),?T(y))
?T(mm)_rep(?T(mm),rep(11,11))
?T(ss)_rep(?T(ss),11)
?T(zz)_?dbinorm(?T(mm),?T(ss),?T(mm),?T(ss),?T(r),?T(x),?T(y))
?T(zz)_?T(zz)/max(?T(zz))
?T(zz)_matrix(?T(zz),11,11,byrow=TRUE)
persp(?T(zz))
if( (min(?T(mm))<=?T(em)) & (max(?T(mm))>=?T(em)) &
(min(?T(ss))<=?T(es)) & (max(?T(ss))>=?T(es)) )  {
title(main="Maximum Likelihood Estimate for Normal Distribution",
   sub=encode("Sigma(L) [",?T(ls),?T(bs),"] Ds =",?T(ds),
   "Mu(R) [",?T(lm),?T(bm),"] Dm =",?T(dm)))
mtext(side=1,line=1,outer=TRUE,
   encode("Given X :",?T(x)," Y :",?T(y)," R :",?T(r)))
mtext(side=1,line=3,outer=TRUE,
   encode("When Mu =",?T(em)," Sigma =",?T(es)," L(M,S) =",?T(mz)))
  }
if( (min(?T(mm))>?T(em)) | (max(?T(mm))<?T(em)) |
(min(?T(ss))>?T(es)) | (max(?T(ss))<?T(es)) )  {
title(main=encode("For given X :",?T(x)," Y :",?T(y)," R :",?T(r)),
   sub=encode("Sigma(L) [",?T(ls),?T(bs),"] Ds =",?T(ds),
   "Mu(R) [",?T(lm),?T(bm),"] Dm =",?T(dm)))
mtext(side=1,line=1,outer=TRUE,
   encode("There is no Mu or Sigma, which will make MLE"))
mtext(side=1,line=3,outer=TRUE,
"If you want to see the MLE for Mu and Sigma, hit Return
 when GO? appears. If not, hit the Break key")
?T(sm)_seq(0,1.5*?T(es),len=12)
?T(sm)_?T(sm)[?T(sm)>0]
?T(ls)_?T(sm)[1]
?T(bs)_?T(sm)[11]
?T(ds)_?T(sm)[2]-?T(sm)[1]
?T(fm)_seq(?T(em)-1.5*?T(es),?T(em)+1.5*?T(es),len=11)
?T(lm)_?T(fm)[1]
?T(bm)_?T(fm)[11]
?T(dm)_?T(fm)[2]-?T(fm)[1]
?T(fm)_rep(?T(fm),rep(11,11))
?T(sm)_rep(?T(sm),11)
?T(zt)_?dbinorm(?T(fm),?T(sm),?T(fm),?T(sm),?T(r),?T(x),?T(y))
?T(zt)_?T(zt)/max(?T(zt))
?T(zt)_matrix(?T(zt),11,11,byrow=TRUE)
par(mfrow=c(1,1), oma=c(7,0,0,0))
persp(?T(zt))
title(main="Maximum Likelihood Estimate for Normal Distribution",
   sub=encode("Sigma(L) [",?T(ls),?T(bs),"] Ds =",?T(ds),
   "Mu(R) [",?T(lm),?T(bm),"] Dm =",?T(dm)))
```

```
    mtext(side=1,line=1,outer=TRUE,
       encode("Given X :",?T(x)," Y :",?T(y)," R :",?T(r)))
    mtext(side=1,line=3,outer=TRUE,
       encode("When Mu =",?T(em)," Sigma =",?T(es)," L(M,S) =",?T(mz)))
    rm(?T(zt),?T(fm),?T(sm))
    }
    rm(?T(em),?T(xm),?T(ym),?T(es),?T(mm),?T(ss),?T(zz),?T(lm),
       ?T(bm),?T(ls),?T(bs),?T(ds),?T(dm))
    }
    rm(?T(x),?T(y),?T(r),?T(m),?T(xx),?T(yy),?T(z),?T(iz),?T(sx),
       ?T(bx),?T(sy),?T(by),?T(dx),?T(dy))
})
END
```

```
MACRO mlelnorm
({
  item_c("Sample size 1",
         "Sample size 2",
         "Go to Previous Step")
  action_c("?mlelnorm1",
           "?mlelnorm2",
           "?mlecont")
  menu( item, action)
  rm( item, action)
})
END



MACRO mlelnorm1
({
  item_c("1 Parameter for Mean",
         "1 Parameter for Sigma",
         "Go to previous step")
  action_c("?mlelnorm1m",
           "?mlelnorm1s",
           "?mlelnorm")
  menu( item , action)
  rm( item, action)
})
END



MACRO mlelnorm2
({
  item_c("1 Parameter for Mean",
         "1 Parameter for Sigma",
         "2 Parameters",
         "Go to previous step")
  action_c("?mlelnorm2m",
           "?mlelnorm2s",
           "?mlelnorm2b",
           "?mlelnorm")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO mlelnorm1m(
x/?PROMPT(Random Sample X ( X > 0 ) : )/,
s/?PROMPT(Given Std Dev             : )/,
m/?PROMPT(Estimated Mean            : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for mean of Lognormal distribution.(Sample size one)
#
({
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    ?T(x)_x
    ?T(s)_s
    ?T(m)_m
    ?T(xx)_seq(0,2*?T(x),len=51)
    ?T(xx)_?T(xx)[?T(xx)>0]
    ?T(y)_dlnorm(?T(xx),?T(m),?T(s))
    ?T(iy)_dlnorm(?T(x),?T(m),?T(s))
    plot(?T(xx),?T(y),type="l",xlab="X-value",ylab="",
       main="PDF for Lognormal Distribution",
       sub=encode("For Given X :",?T(x)," When Mean =",?T(m),
       " Std Dev =",?T(s)," PDF =",?T(iy)),
       cex=.5,err=-1,ylim=c(0,max(?T(y))))
    mtext(side=2, line=1, outer=T, "PDF")
    lines(c(?T(x),?T(x),0),c(0,?T(iy),?T(iy)))
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for Mean, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(mm)_log(?T(x))
    ?T(em)_seq(?T(mm)-2*?T(s),?T(mm)+2*?T(s),len=51)
    ?T(yy)_dlnorm(rep(?T(x),len(?T(em))),?T(em),?T(s))
    ?T(my)_dlnorm(?T(x),?T(mm),?T(s))
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    plot(?T(em),?T(yy),type="l",xlab="Mean",ylab="",
       main="Maximum Likelihood for Lognormal Distribution",
       sub=encode("For Given X :",?T(x)," S :",?T(s),
       " When M =",?T(mm)," L(M) =",?T(my)),ylim=c(0,1.1*?T(my)))
    mtext(side=2, line=1, outer=T, "MLE")
    sm.x_min(?T(em))
    sm.x_ifelse(sm.x>0, ceiling(sm.x),floor(sm.x))
    lines(c(?T(m),?T(m),sm.x),c(0,?T(iy),?T(iy)))
    arrows(?T(mm),?T(my),?T(mm),0)
    arrows(?T(mm),?T(my),sm.x,?T(my))
    rm(?T(x),?T(s),?T(m),?T(xx),?T(em),?T(y),?T(iy),?T(mm),
       ?T(yy),?T(my),sm.x)
})
END
```

```
MACRO mlelnorm1s(
x/?PROMPT(Random Sample X       : )/,
m/?PROMPT(Given Mu              : )/,
s/?PROMPT(Estimated Sigma (S>0) : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Lognormal distribution( For one sample with known Mu).
#
({
    ?T(x)_x
    ?T(m)_m
    ?T(s)_s
    if(log(?T(x))==?T(m)) {
    print ("When Log(X) and Mu are same,")
    print ("the MLE Sigma is zero")
    }
    if(log(?T(x))!=?T(m)) {
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    ?T(ms)_abs(log(?T(x))-?T(m))
    ?T(xx)_seq(0,?T(x)+2*?T(ms),len=51)
    ?T(xx)_?T(xx)[?T(xx)>0]
    ?T(iy)_dlnorm(?T(x),?T(m),?T(s))
    ?T(l)_dlnorm(?T(xx),?T(m),?T(s))
    plot(?T(xx),?T(l),type="l",
          xlab="X",ylab="",
          main="PDF for Lognormal Distribution",
          sub=encode("For Given X :",?T(x)," Mu :",?T(m),
          " When S =",?T(s)," PDF =",?T(iy)))
    lines(c(?T(x),?T(x),0),c(0,?T(iy),?T(iy)))
    mtext(side=2, line=1, outer=T, "PDF")
    mtext(side=1,line=1,outer=TRUE,
    "If you want to see the MLE for Sigma, hit Return
    when GO? appears.  If not, hit the Break key")
    ?T(es)_max(?T(ms),?T(s))
    ?T(es)_seq(0,1.5*?T(es),len=51)
    ?T(es)_?T(es)[?T(es)>0]
    ?T(ns)_len(?T(es))
    ?T(y)_dlnorm(rep(?T(x),?T(ns)),rep(?T(m),?T(ns)),?T(es))
    ?T(ml)_dlnorm(?T(x),?T(m),?T(ms))
    plot(?T(es),?T(y),type="l",
          xlab="Sigma",ylab="",ylim=c(0,1.1*?T(ml)),
          main="Maximum Likelihood Estimate for Lognormal Distribution",
          sub=encode("Given X :",?T(x),"Mu :",?T(m),"When S =",?T(ms),
          "L(S) =",?T(ml)),cex=.5,err=-1)
    mtext(side=2, line=1, outer=T, "MLE")
    lines(c(?T(s),?T(s),0),c(0,?T(iy),?T(iy)))
    arrows(?T(ms),?T(ml),?T(ms),0)
    arrows(?T(ms),?T(ml),0,?T(ml))
    rm(?T(x),?T(m),?T(s),?T(xx),?T(iy),?T(l),?T(ms),
       ?T(es),?T(ns),?T(ml),?T(y))
    }
})
END
```

```
MACRO mlelnorm2m(
x/?PROMPT(Random Sample X ( X > 0 ) : )/,
y/?PROMPT(Random Sample Y ( Y > 0 ) : )/,
s/?PROMPT(Given Std Dev             : )/,
m/?PROMPT(Estimated Mean            : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for mean of Lognormal distribution.(Sample size two)
#
({
    par(mfrow=c(1,1), oma=c(7,1,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(s)_s
    ?T(m)_m
    ?T(xx)_seq(0,2*?T(x),len=11)
    ?T(yy)_seq(0,2*?T(y),len=11)
    ?T(xx)_?T(xx)[?T(xx)>0]
    ?T(sx)_?T(xx)[1]
    ?T(bx)_?T(xx)[10]
    ?T(dx)_?T(xx)[2]-?T(xx)[1]
    ?T(yy)_?T(yy)[?T(yy)>0]
    ?T(sy)_?T(yy)[1]
    ?T(by)_?T(yy)[10]
    ?T(dy)_?T(yy)[2]-?T(yy)[1]
    ?T(xz)_dlnorm(?T(xx),?T(m),?T(s))
    ?T(yz)_dlnorm(?T(yy),?T(m),?T(s))
    ?T(xz)_matrix(?T(xz),10,1)
    ?T(yz)_matrix(?T(yz),1,10)
    ?T(z)_?T(xz)%*?T(yz)
    ?T(z)_?T(z)/max(?T(z))
    ?T(iz)_dlnorm(?T(x),?T(m),?T(s))*dlnorm(?T(y),?T(m),?T(s))
    persp(?T(z))
    title(main="PDF for Lognormal Distribution",
       sub=encode("Y(L) : [",?T(sy),?T(by),"] Dy =",?T(dy),
       " X(R) : [",?T(sx),?T(bx),"] Dx =",?T(dx)),err=-1)
    mtext(side=1,line=1,cex=2,outer=TRUE,
       encode("For Given X :",?T(x)," Y :",?T(y)," When Mu =",
       ?T(m)," S =",?T(s)," PDF =",?T(iz)))
    mtext(side=1,line=3,cex=2,outer=TRUE,
    "If you want to see the MLE for Mean, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(mm)_mean(log(c(?T(x),?T(y))))
    ?T(em)_seq(?T(mm)-2*?T(s),?T(mm)+2*?T(s),len=51)
    ?T(yy)_dlnorm(rep(?T(x),len(?T(em))),?T(em),?T(s))*
           dlnorm(rep(?T(y),len(?T(em))),?T(em),?T(s))
    ?T(my)_dlnorm(?T(x),?T(mm),?T(s))*dlnorm(?T(y),?T(mm),?T(s))
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    plot(?T(em),?T(yy),type="l",xlab="Mean",ylab="",
       main="Maximum Likelihood for Lognormal Distribution",
       sub=encode("For Given X :",?T(x)," Y :",?T(y)," S :",?T(s),
       " When M =",?T(mm)," L(M) =",?T(my)),ylim=c(0,1.1*?T(my)),
```

```
      cex=.5,err=-1)
   mtext(side=2, line=1, outer=T, "MLE")
   sm.x_min(?T(em))
   sm.x_ifelse(sm.x>0, ceiling(sm.x),floor(sm.x))
   lines(c(?T(m),?T(m),sm.x),c(0,?T(iz),?T(iz)))
   arrows(?T(mm),?T(my),?T(mm),0)
   arrows(?T(mm),?T(my),sm.x,?T(my))
   rm(?T(x),?T(s),?T(m),?T(xx),?T(em),?T(y),?T(iz),?T(mm),
      ?T(my),?T(yy),?T(xz),?T(yz),?T(z),?T(sx),?T(bx),?T(dx),
      ?T(sy),?T(by),?T(dy),sm.x)
})
END
```

```
MACRO mlelnorm2s(
x/?PROMPT(Random Sample X ( X > 0 ) : )/,
y/?PROMPT(Random Sample Y ( Y > 0 ) : )/,
m/?PROMPT(Given Mu                  : )/,
s/?PROMPT(Estimated Std Dev         : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for sigma of Lognormal distribution.(Sample size two)
#
({
    par(mfrow=c(1,1), oma=c(7,3,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(m)_m
    ?T(s)_s
    ?T(xx)_seq(0,2*?T(x),len=11)
    ?T(yy)_seq(0,2*?T(y),len=11)
    ?T(xx)_?T(xx)[?T(xx)>0]
    ?T(sx)_?T(xx)[1]
    ?T(bx)_?T(xx)[10]
    ?T(dx)_?T(xx)[2]-?T(xx)[1]
    ?T(yy)_?T(yy)[?T(yy)>0]
    ?T(sy)_?T(yy)[1]
    ?T(by)_?T(yy)[10]
    ?T(dy)_?T(yy)[2]-?T(yy)[1]
    ?T(xz)_dlnorm(?T(xx),?T(m),?T(s))
    ?T(yz)_dlnorm(?T(yy),?T(m),?T(s))
    ?T(xz)_matrix(?T(xz),10,1)
    ?T(yz)_matrix(?T(yz),1,10)
    ?T(z)_?T(xz)%*?T(yz)
    ?T(z)_?T(z)/max(?T(z))
    ?T(iz)_dlnorm(?T(x),?T(m),?T(s))*dlnorm(?T(y),?T(m),?T(s))
    persp(?T(z))
    title(main="PDF for Lognormal Distribution",
       sub=encode("Y(L) : [",?T(sy),?T(by),"] Dy =",?T(dy),
       " X(R) : [",?T(sx),?T(bx),"] Dx =",?T(dx)),err=-1)
    mtext(side=1,line=1,cex=2,outer=TRUE,
       encode("For Given X :",?T(x)," Y :",?T(y)," When Mu =",
       ?T(m)," S =",?T(s)," PDF =",?T(iz)))
    if( ?T(x)==?T(y) & log(?T(x))==?T(m) ) {
    mtext(side=1,line=3,cex=2,outer=TRUE,
     "When X = Y and Log(X) is same as Given Mean
     MLE for Std Dev = 0")
     }     else {
    mtext(side=1,line=3,cex=2,outer=TRUE,
    "If you want to see the MLE for Std Dev, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(xy)_log(c(?T(x),?T(y)))
    ?T(ms)_sqrt(sum((?T(m)-?T(xy))^2)/2)
    ?T(es)_max(?T(s),?T(ms))
    ?T(es)_seq(0,2*?T(es),len=51)
    ?T(es)_?T(es)[?T(es)>0]
```

```
    ?T(yy)_dlnorm(rep(?T(x),len(?T(es))),?T(m),?T(es))*
          dlnorm(rep(?T(y),len(?T(es))),?T(m),?T(es))
    ?T(my)_dlnorm(?T(x),?T(m),?T(ms))*dlnorm(?T(y),?T(m),?T(ms))
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    plot(?T(es),?T(yy),type="l",xlab="Std Dev",ylab="",
       main="Maximum Likelihood for Lognormal Distribution",
       ylim=c(0,1.1*?T(my)), cex=.5,err=-1)
    mtext(side=2, line=1, outer=T, "MLE")
    mtext(side=1, line=1, outer=T,
       encode("For Given X :",?T(x)," Y :",?T(y)," Mu :",?T(m)))
    mtext(side=1, line=3, outer=T,
       encode(" When S =",?T(ms)," L(M) =",?T(my)))
    lines(c(?T(s),?T(s),0),c(0,?T(iz),?T(iz)))
    arrows(?T(ms),?T(my),?T(ms),0)
    arrows(?T(ms),?T(my),0,?T(my))
    rm(?T(x),?T(s),?T(m),?T(xx),?T(es),?T(y),?T(iz),?T(ms),
       ?T(my),?T(yy),?T(xz),?T(yz),?T(z),?T(sx),?T(bx),
       ?T(dx),?T(sy),?T(by),?T(dy))
   }
})
END
```

```
MACRO mlelnorm2b(
x/?PROMPT(Random Sample X          :)/,
y/?PROMPT(Random Sample Y          :)/,
m/?PROMPT(Estimated Mu             :)/,
s/?PROMPT(Estimated Sigma ( S>0 ) :)/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Lognormal distribution( For two sample with unknown Mu & S).
#
({
    ?T(x)_x
    ?T(y)_y
    ?T(m)_m
    ?T(s)_s
    par(mfrow=c(1,1), oma=c(6,0,0,0))
    ?T(xx)_seq(0,2*?T(x),len=11)
    ?T(yy)_seq(0,2*?T(y),len=11)
    ?T(xx)_?T(xx)[?T(xx)>0]
    ?T(sx)_?T(xx)[1]
    ?T(bx)_?T(xx)[10]
    ?T(dx)_?T(xx)[2]-?T(xx)[1]
    ?T(yy)_?T(yy)[?T(yy)>0]
    ?T(sy)_?T(yy)[1]
    ?T(by)_?T(yy)[10]
    ?T(dy)_?T(yy)[2]-?T(yy)[1]
    ?T(xz)_dlnorm(?T(xx),?T(m),?T(s))
    ?T(yz)_dlnorm(?T(yy),?T(m),?T(s))
    ?T(xz)_matrix(?T(xz),10,1)
    ?T(yz)_matrix(?T(yz),1,10)
    ?T(z)_?T(xz)%*?T(yz)
    ?T(z)_?T(z)/max(?T(z))
    ?T(iz)_dlnorm(?T(x),?T(m),?T(s))*dlnorm(?T(y),?T(m),?T(s))
    persp(?T(z))
    title(main="PDF for Lognormal Distribution",
      sub=encode("Y(L) : [",?T(sy),?T(by),"] Dy =",?T(dy),
      " X(R) : [",?T(sx),?T(bx),"] Dx =",?T(dx)),err=-1)
    mtext(side=1,line=1,cex=2,outer=TRUE,
      encode("For Given X :",?T(x)," Y :",?T(y)))
    mtext(side=1,line=2,cex=2,outer=TRUE,
      encode(" When Mu =", ?T(m)," S =",?T(s)," PDF =",?T(iz)))
    mtext(side=1,line=4,cex=2,outer=TRUE,
    "If you want to see the MLE for Mu & S, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(xy)_log(c(?T(x),?T(y)))
    ?T(mm)_mean(?T(xy))
    if(?T(x)!=?T(y)) ?T(ms)_sqrt(sum((?T(mm)-?T(xy))^2)/2) else
                        ?T(ms)_.01
#
# If the case of that X = Y, the MLE for mean is mean of log(x) and
# log(y), and then the MLE for sigma should be 0. Since, however,
# 0 sigma can't be plotted, minimum sigma is assummed .01.
#
```

```
    par(mfrow=c(1,1), oma=c(6,0,0,0))
    ?T(em)_seq(?T(mm)-2*?T(ms),?T(mm)+2*?T(ms),len=11)
    ?T(es)_seq(0,3*?T(ms),len=11)
    ?T(em)_?T(em)[2:11]
    ?T(dm)_?T(em)[2]-?T(em)[1]
    ?T(es)_?T(es)[2:11]
    ?T(ds)_?T(es)[2]-?T(es)[1]
    ?T(rm)_rep(?T(em),rep(10,10))
    ?T(rs)_rep(?T(es),10)
    ?T(xl)_dlnorm(rep(?T(x),100),?T(rm),?T(rs))
    ?T(yl)_dlnorm(rep(?T(y),100),?T(rm),?T(rs))
    ?T(l)_?T(xl)*?T(yl)
    ?T(l)_?T(l)/max(?T(l))
    ?T(l)_matrix(?T(l),10,10,byrow=TRUE)
    ?T(il)_dlnorm(?T(x),?T(mm),?T(ms))*dlnorm(?T(y),?T(mm),?T(ms))
    persp(?T(l))
    title(main="Maximum Likelihood Estimate for Lognormal Distribution",
      sub=encode("S(L):[",?T(es)[1],?T(es)[10],"]",
      " Mu(R):[",?T(em)[1],?T(em)[10],"]"))
    mtext(side=1,line=1,outer=TRUE,
      encode("Increment of S :",?T(ds),
      "       Increment of Mu :",?T(dm)))
    if(?T(x)!=?T(y)) {
    mtext(side=1,line=3,outer=TRUE,
      encode("Given X:",?T(x),"Y:",?T(y)))
    mtext(side=1,line=4,outer=TRUE,
      encode(" When Mu=",?T(mm),"S=",?T(ms)," L(M,S)=",?T(il)))
      } else {
    mtext(side=1,line=1,outer=TRUE,
      encode("Given X = Y =",?T(y)," Mu=",?T(mm),
      "S= 0 will be the MLE"))
    }
    rm(?T(x),?T(y),?T(m),?T(s),?T(xx),?T(yy),?T(xz),?T(yz),?T(z),
       ?T(iz),?T(xy),?T(mm),?T(em),?T(ms),?T(es),?T(xl),?T(yl),
       ?T(l),?T(il),?T(rm),?T(rs))
})
END
```

```
MACRO mleexpon
({
  item_c("Sample size 1",
         "Sample size 2",
         "Go to Previous Step")
  action_c("?mleexp1",
           "?mleexp2",
           "?mlecont")
  menu( item , action)
  rm( item, action)
})
END


MACRO mleexp1(
x/?PROMPT(Random Sample X  : )/,
lm/?PROMPT(Estimated Lambda : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Exponential distribution.
#
({
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    ?T(x)_x
    ?T(l)_lm
    ?T(xx)_seq(0,2*?T(x),len=51)
    ?T(y)_?dex1(?T(xx),?T(l))
    ?T(iy)_?dex1(?T(x),?T(l))
    plot(?T(xx),?T(y),type="l",xlab="X-value",ylab="",
       main="PDF for Exponential Distribution",
       sub=encode("For Given X :",?T(x)," When Lambda =",?T(l),
       "PDF =",?T(iy)),cex=.5,err=-1,ylim=c(0,max(?T(y))))
    lines(c(?T(x),?T(x),0),c(0,?T(iy),?T(iy)))
    mtext(side=2, line=1, outer=T, "PDF")
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for Lambda, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(ll)_seq(0,2/?T(x),len=51)
    ?T(yy)_?dex1(?T(x),?T(ll))
    ?T(my)_?dex1(?T(x),1/?T(x))
    par(mfrow=c(1,1), oma=c(0,3,0,0))
    plot(?T(ll),?T(yy),type="l",xlab="Lambda",ylab="",
       main="Maximum Likelihood for Exponential Distribution",
       sub=encode("Given X :",?T(x)," When L =",1/?T(x),
       "L(L) =",?T(my)),cex=.5,err=-1,ylim=c(0,1.1*?T(my)))
    mtext(side=2, line=1, outer=T, "MLE")
    lines(c(?T(l),?T(l),0),c(0,?T(iy),?T(iy)))
    arrows(1/?T(x),?T(my),1/?T(x),0)
    arrows(1/?T(x),?T(my),0,?T(my))
    rm(?T(x),?T(l),?T(xx),?T(y),?T(iy),?T(ll),?T(yy),?T(my))
})
END
```

```
MACRO mleexp2(
x/?PROMPT(Random Sample X  : )/,
y/?PROMPT(Random Sample Y  : )/,
lm/?PROMPT(Estimated Lambda : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Exponential distribution.(Sample size two)
#
({
    par(mfrow=c(1,1), oma=c(7,0,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(l)_lm
    ?T(xx)_seq(0,2*(max(?T(x),?T(y))),len=11)
    ?T(d)_?T(xx)[2]-?T(xx)[1]
    ?T(xz)_?dex1(?T(xx),?T(l))
    ?T(xz)_matrix(?T(xz),11,1)
    ?T(yz)_t(?T(xz))
    ?T(z)_?T(xz)%*?T(yz)
    ?T(z)_?T(z)/max(?T(z))
    ?T(iz)_?dex1(?T(x),?T(l))*?dex1(?T(y),?T(l))
    persp(?T(z))
    title(main="PDF for Exponential Distribution",
       encode("X(L) = Y(R) : [ 0",?T(xx)[11],"]   Increment :",
       ?T(d)))
    mtext(side=1,line=1,cex=2,outer=TRUE,
       encode("Given X :",?T(x)," Y :",?T(y),
       " When Lambda =",?T(l),"PDF =",?T(iz)))
    mtext(side=1,line=4,cex=2,outer=TRUE,
    "If you want to see the MLE for Lambda, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(ml)_2/(?T(x)+?T(y))
    ?T(ll)_seq(0,2*?T(ml),len=51)
    ?T(yy)_?dex1(?T(x),?T(ll))*?dex1(?T(y),?T(ll))
    ?T(my)_?dex1(?T(x),?T(ml))*?dex1(?T(y),?T(ml))
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    plot(?T(ll),?T(yy),type="l",xlab="Lambda",ylab="",
       main="Maximum Likelihood for Exponential Distribution",
       cex=.5,err=-1,ylim=c(0,1.1*?T(my)))
    mtext(side=2, line=1, outer=T, "MLE")
    mtext(side=1,line=0,cex=.5,outer=TRUE,
       encode("Given X:",?T(x)," Y:",?T(y)," When L =",
       ?T(ml)," L(L) =",?T(my)))
    lines(c(?T(l),?T(l),0),c(0,?T(iz),?T(iz)))
    arrows(?T(ml),?T(my),?T(ml),0)
    arrows(?T(ml),?T(my),0,?T(my))
    rm(?T(x),?T(y),?T(l),?T(xx),?T(xz),?T(yz),?T(z),?T(iz),?T(ml),
       ?T(ll),?T(yy),?T(my),?T(d))
})
END
```

```
MACRO mlecunif(
x/?PROMPT(Random Sample X ( Any Integer ) : )/,
y/?PROMPT(Random Sample Y ( Any Integer ) : )/,
l/?PROMPT(Estimated Lower Bound           : )/,
u/?PROMPT(Estimated Upper Bound ( U > L ) : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Continuous Uniform distribution with unknown parameter.
#
({
    par(mfrow=c(1,1), oma=c(5,0,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(rs)_c(?T(x),?T(y))
    ?T(l)_l
    ?T(u)_u
    ?T(bs)_max(?T(rs))
    ?T(ss)_min(?T(rs))
    ?T(mx)_max(?T(bs),?T(u)) + 2
    ?T(mn)_min(?T(ss),?T(l)) - 2
    ?T(px)_seq(?T(mx),?T(mn),len=10)
    ?T(pmf)_1/(?T(u)-?T(l))
# PMF for discrete uniform distribution is;
#     pmf = 1/(upper - lower +1)
    ?T(px)_ifelse( (?T(px)<=?T(u) & ?T(px)>=?T(l) ),?T(pmf),0)
    ?T(px)_matrix(?T(px),ncol=1)
    ?T(py)_t(?T(px))
    ?T(z)_?T(px)%*?T(py)
# Joint pmf for independent uniform distribution can be computed
# by the matrix form.
    ?T(pmf)_?T(pmf)^2
    ?T(z)_?T(z)/?T(pmf)
    persp(?T(z))
    ?T(pmf)_ifelse(?T(l)<=?T(ss) & ?T(u)>=?T(bs), ?T(pmf), 0)
    title(main="PMF for Continuous Uniform Distribution",
       sub=encode("For Given Samples",?T(ss),?T(bs),
       "When Boundary = (", ?T(l),?T(u),")  PMF =",?T(pmf)))
    if(?T(x)==?T(y)) {
    print ("When X = Y, Lower Bound = Upper Bound at X or Y") }
    if(?T(x)!=?T(y)) {
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for Upper/Lower Bound,
    hit Return when GO? appears.  If not, hit the Break key")
    ?T(u)_seq(?T(bs),?T(bs)+3,len=11)
    ?T(l)_seq(?T(ss)-3,?T(ss),len=11)
# Select the range of Upper and Lower Bound
    ?T(u)_rep(?T(u),rep(11,11))
    ?T(l)_rep(?T(l),11)
    ?T(pmf)_(1/(?T(u)-?T(l)))^2
    ?T(ml)_max(?T(pmf))
    ?T(zz)_matrix(?T(pmf),11,11,byrow=TRUE)
    ?T(zz)_?T(zz)/?T(ml)
```

```
    par(mfrow=c(1,1), oma=c(5,0,0,0))
    persp(?T(zz))
    title(main="Maximum Likelihood for Continuous Uniform Distribution",
       sub=encode("For Given Samples",?T(ss),?T(bs)),cex=.5,err=-1)
    mtext(side=1,line=1,outer=TRUE,
       encode("When Boundary = (",?T(ss),?T(bs),")  L(P) =",?T(ml)))
    rm(?T(ml),?T(zz))
       }
    rm(?T(x),?T(y),?T(l),?T(u),?T(rs),?T(bs),?T(ss),?T(mx),
       ?T(mn),?T(px),?T(pmf),?T(py),?T(z))
})
END
```

```
MACRO mlestgam
({
  item_c("Sample size 1",
         "Sample size 2",
         "Go to Previous Step")
  action_c("?mlestgam1",
           "?mlestgam2",
           "?mlecont")
  menu( item , action)
  rm( item, action)
})
END
```

```
MACRO mlestgam1(
x/?PROMPT(Random Sample X ( X > 0 ) : )/,
al/?PROMPT(Estimated Shape Parameter : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Standard Gamma distribution.(Sample size one)
#
({
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    ?T(x)_x
    ?T(a)_al
    ?T(xx)_seq(0,2*?T(x),len=101)
    ?T(xx)_?T(xx)[?T(xx)>0]
    ?T(y)_dgamma(?T(xx),?T(a))
    ?T(iy)_dgamma(?T(x),?T(a))
    plot(?T(xx),?T(y),type="l",xlab="X-value",ylab="",
       main="PDF for Standard Gamma Distribution",
       sub=encode("For Given X :",?T(x)," When Alpha =",?T(a),
       "PDF =",?T(iy)),cex=.5,err=-1,ylim=c(0,max(?T(y))))
    mtext(side=2, line=1, outer=T, "PDF")
    lines(c(?T(x),?T(x),0),c(0,?T(iy),?T(iy)))
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for Alpha, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(ea)_max(?T(a),?T(x))
    ?T(ea)_seq(0,2*?T(ea),len=101)
    ?T(k)_0
#
# Repeatation will be made until reasonable precise MLE for
# Shape parameter can be obtained. Since it is not possible to
# compute the MLE parameter, the repeatation by the graphic is
# done.
#
  for (i in 1:10) {
    ?T(k)_?T(k)+1
    ?T(ea)_?T(ea)[?T(ea)>0]
    ?T(yy)_dgamma(rep(?T(x),len(?T(ea))),?T(ea))
    ?T(sa)_min(?T(ea))
    ?T(ba)_max(?T(ea))
    ?T(sy)_min(?T(yy))
    ?T(my)_max(?T(yy))
    ?T(po)_(1:len(?T(yy)))[?T(yy)==?T(my)]
    ?T(ma)_?T(ea)[?T(po)]
    par(mfrow=c(1,1), oma=c(5,3,0,0))
    plot(?T(ea),?T(yy),type="l",xlab="Alpha",ylab="",
       main="Maximum Likelihood for Standard Gamma Distribution",
       sub=encode("For Given X :",?T(x)," When A =",?T(ma),
       "L(A) =",?T(my)),cex=.5,err=-1,ylim=c(?T(sy),?T(my)),
       xlim=c(?T(sa),?T(ba)))
    mtext(side=2, line=1, outer=T, "MLE")
     if (?T(k)==1) {
      lines(c(?T(a),?T(a),0),c(0,?T(iy),?T(iy)))
```

```
      }
    arrows(?T(ma),?T(my),?T(ma),?T(sy))
    arrows(?T(ma),?T(my),?T(sa),?T(my))
    ?T(ea)_seq(?T(ea)[max(1,?T(po)[1]-5)],
    ?T(ea)[min(101,?T(po)[len(?T(po))]+5)],len=101)
#
# To divide the interval more precisely, the range will be
# narrowed down.
#
    if (?T(k)<10) {
      mtext(side=1,line=1,cex=2,outer=TRUE,
      "If you want to see the MLE for Alpha more precisely,
      hit Return when GO? appears.  If not, hit the Break key")
      }
    rm(?T(yy),?T(sa),?T(ba),?T(sy),?T(my),?T(po),?T(ma))
    }
    rm(?T(x),?T(a),?T(xx),?T(y),?T(iy),?T(ea),?T(k))
})
END
```

```
MACRO mlestgam2(
x/?PROMPT(Random Sample X ( X > 0 ) : )/,
y/?PROMPT(Random Sample Y ( Y > 0 ) : )/,
al/?PROMPT(Estimated Shape Parameter : )/)
#
# This is a Macro to plot the maximum likelihood estimate
# for Standard Gamma distribution.(Sample size two)
#
({
    par(mfrow=c(1,1), oma=c(5,0,0,0))
    ?T(x)_x
    ?T(y)_y
    ?T(a)_al
    ?T(xx)_seq(0,2*?T(x),len=11)
    ?T(xx)_?T(xx)[?T(xx)>0]
    ?T(yy)_seq(0,2*?T(y),len=11)
    ?T(yy)_?T(yy)[?T(yy)>0]
    ?T(xz)_dgamma(?T(xx),?T(a))
    ?T(yz)_dgamma(?T(yy),?T(a))
    ?T(xz)_matrix(?T(xz),10,1)
    ?T(yz)_matrix(?T(yz),1,10)
    ?T(z)_?T(xz)%*?T(yz)
    ?T(z)_?T(z)/max(?T(z))
    ?T(iz)_dgamma(?T(x),?T(a))*dgamma(?T(y),?T(a))
    persp(?T(z))
    title(main="PDF for Standard Gamma Distribution",
      sub=encode("For Given X :",?T(x),"Y :",?T(y)," When Alpha =",
      ?T(a),"PDF =",?T(iz)))
    mtext(side=1,line=1,cex=2,outer=TRUE,
    "If you want to see the MLE for Alpha, hit Return
     when GO? appears.  If not, hit the Break key")
    ?T(ea)_max(?T(a),(?T(x)+?T(y))/2)
    ?T(ea)_seq(0,2*?T(ea),len=101)
    ?T(k)_0
#
# Repeatation will be made until reasonable precise MLE for
# Shape parameter can be obtained. Since it is not possible to
# compute the MLE parameter, the repeatation by the graphic is
# done.
#
    for ( i in 1:10) {
    ?T(k)_?T(k)+1
    ?T(ea)_?T(ea)[?T(ea)>0]
    ?T(ly)_dgamma(rep(?T(x),len(?T(ea))),?T(ea))*
           dgamma(rep(?T(y),len(?T(ea))),?T(ea))
    ?T(sa)_min(?T(ea))
    ?T(ba)_max(?T(ea))
    ?T(sy)_min(?T(ly))
    ?T(my)_max(?T(ly))
    ?T(po)_(1:len(?T(ly)))[?T(ly)==?T(my)]
    ?T(ma)_?T(ea)[?T(po)]
    par(mfrow=c(1,1), oma=c(5,3,0,0))
```

```
    plot(?T(ea),?T(ly),type="l",xlab="Alpha",ylab="",
       main="Maximum Likelihood for Standard Gamma Distribution",
       sub=encode("Given X :",?T(x)," Y :",?T(y)," When A =",
       ?T(ma),"L(A) =",?T(my)),cex=.5,err=-1,ylim=c(?T(sy),?T(my)),
       xlim=c(?T(sa),?T(ba)))
    mtext(side=2, line=1, outer=T, "MLE")
    if (?T(k)==1) {
      lines(c(?T(a),?T(a),0),c(0,?T(iz),?T(iz)))
      }
    arrows(?T(ma),?T(my),?T(ma),?T(sy))
    arrows(?T(ma),?T(my),?T(sa),?T(my))
    ?T(ea)_seq(?T(ea)[max(1,?T(po)[1]-5)],
    ?T(ea)[min(101,?T(po)[len(?T(po))]+5)],len=101)
#
# To divide the interval more precisely, the range will be
# narrowed down.
#
    if (?T(k)<10) {
      mtext(side=1,line=1,cex=2,outer=TRUE,
      "If you want to see the MLE for Alpha more precisely,
      hit Return when GO? appears.  If not, hit the Break key")
      }
    rm(?T(ly),?T(sa),?T(ba),?T(sy),?T(my),?T(po),?T(ma))
    }
    rm(?T(x),?T(a),?T(xx),?T(y),?T(yy),?T(iz),?T(ea),?T(k),
       ?T(xz),?T(yz),?T(z))
})
END
```

Appendix C: Macros for Transformation of Random Variables

Table of Contents

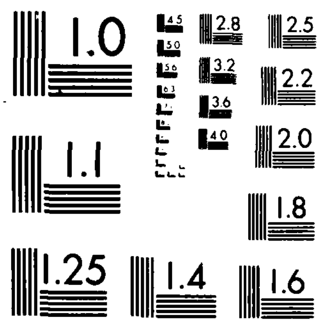
Page

```
MACRO tran
#
# This MAcro is to select univariate or bivariate
# distribution for transformation.
#
({
    item_c("Univariate distribution",
           "Bivariate  distribution")
    action_c("?uvtran",
             "?bvtran")
   menu( item, action)
   rm( item, action)
})
END
```

```
MACRO uvtran
({
  ?MESSAGE()
  ?MESSAGE(Type of Distribution :)
  ?MESSAGE()
  item_c("Discrete   Distribution",
         "Continuous Distribution")
  action_c("?trbin",
           "?trcont")
  menu( item, action)
  rm( item, action)
})
END
```

AD-A174 297 DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL PACKAGE (ISTP) FOR LE..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF SYST.. K H CHUL 4/5

UNCLASSIFIED SEP 86 AFIT/GSM/ENC/86S-10 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
MACRO trcont
({
  ?MESSAGE()
  ?MESSAGE(Type of Distribution :)
  ?MESSAGE()
   item_c("Standard  Normal   Distribution",
          "Normal       Distribution",
          "Uniform      Distribution",
          "Lognormal    Distribution",
          "Exponential  Distribution",
          "Gamma        Distribution",
          "F            Distribution",
          "T            Distribution",
          "Weibull      Distribution",
          "Chisquare    Distribution")
   action_c("?trstnorm",
            "?trnorm",
            "?trunif",
            "?trlnorm",
            "?trexp",
            "?trgamma",
            "?trf",
            "?trt",
            "?trweib",
            "?trchisq")
   menu(item,action)
   rm(item, action)
})
END
```

```
MACRO trbin(
av/?PROMPT(Coefficient of X^2 : )/,
bv/?PROMPT(Coefficient of X   : )/,
cv/?PROMPT(Coef of Constant   : )/,
nv/?PROMPT(Number of Trial    : )/,
pv/?PROMPT(Prob of Success    : )/)
({
  ?MESSAGE()
  ?MESSAGE(Binomial distribution will be transformed)
  ?MESSAGE(with Y = a X^2 + b X + c form. )
  ?MESSAGE(Please enter the transform equation.)
  ?MESSAGE()
#
# Transform the input to the aX^2+bX+x form.
#
  a_av
  as_ifelse(a>=0,"","-")
  b_bv
  bs_ifelse(b>=0,"+","-")
  d_cv
  ds_ifelse(d>=0,"+","-")
  ?MESSAGE()
  ?MESSAGE(Please enter the parameters for Binomial distribution.)
  ?MESSAGE()
#
# Binomial parameters
#
  n_nv
  p_pv
  x_0:n
  trseq_encode("Y =",a,"X^2",bs,abs(b),ds,abs(d))
  y_a*x^2+b*x+d
#
# Compute the pmf for binomial distribution
#
  bin_?mbin(x,n,p)
#
# If same y exists for different x, the pmf for y is
# sum of the pmf for different x.
#
  ym_matrix(y,len(y),len(y),byrow=TRUE)
  ym_ifelse(ym==y,1,0)
  yr_?row(ym,sum)
  ix_?which(yr!=1)
  ybin_bin
  for ( i in ix ) {
    ybin[i]_sum( bin[?which(ym[i,]==1)] )
    }
```

```
# Plot the relationship between x and y
#
  par(mfrow=c(1,2), oma=c(7,0,5,0) )
  plot(x,y,type="h",xlab="X",ylab="Y", err=-1)
  plot(y,x,type="h",xlab="Y",ylab="X", err=-1)
  mtext(side=3,line=0,outer=TRUE,
    "Transform of Binomial Variable")
  mtext(side=1,line=0,outer=TRUE,trseq)
  mtext(side=1,line=2,outer=TRUE,
     "Hit Return for Transform of PMF, when GO? appears")
#
# Plot x vs. pmf
#
  par(oma=c(4,3,3,0))
  plot(x,bin,type="h",xlab="X",ylab="", err=-1,
    sub=encode("Binomial with N :",n,"P :",p),
    xlim=c(-1,max(x)+1),ylim=c(0,max(bin,ybin)*1.01))
#
# Plot y vs. pmf
#
  plot(y,ybin,type="h",xlab="Y",ylab="", sub=trseq, err=-1,
    xlim=c(min(y)-1,max(y)+1),ylim=c(0,max(ybin,bin)*1.01))
  mtext(side=2,line=1,outer=TRUE,"PMF")
  mtext(side=3,line=0,outer=TRUE,
  encode("Transform of Binomial variable  ",trseq))
  mtext(side=1,line=1,outer=TRUE,
  "This is overall picture fof transform of binomial, if you want
  to see the picture one by one, keep going on hitting Return
  when GO? appears until you hit Break key.")
#
# Plot iterative picture
#
  for( i in 1:len(x)) {
  par(oma=c(5,3,5,0))
#
# Plot cumulative x and their pmf
#
  plot(x[1:i],bin[1:i],type="h",xlab="X",ylab="",
    sub=encode("Binomial with N :",n,"P :",p), err=-1,
    xlim=c(-1,x[i]+1),ylim=c(0,max(bin[1:i],ybin[1:i])*1.01))
#
# Indicate the current plot
#
  points(x[i],bin[i])
#
# Plot cumulative y and their pmf
#
  plot(y[1:i],ybin[1:i],type="h",xlab="Y",ylab="",
    sub=encode("Y =",as,abs(a),"X^2",bs,abs(b),"X",ds,abs(d)),
    xlim=c(min(y[1:i])-1,max(y[1:i])+1),
    ylim=c(0,max(ybin[1:i],bin[1:i])*1.01), err=-1)
```

```
#
# Indicate the current plot
#
  points(y[i],ybin[i])
  mtext(side=2,line=1,outer=TRUE,"PMF")
  mtext(side=3,line=1,outer=TRUE,
  encode("Transform of Binomial Distribution  ",trseq))
  mtext(side=1,line=1,outer=TRUE,
  encode("When X =",x[i],"PMF =",bin[i],"  Y =",y[i],
  "PMF =",ybin[i]))
  }
  rm(a,b,d,p,n,y,bin,ybin,x,ym,yr,ix,as,bs,ds,trseq)
})
END
```

```
MACRO trstnorm(av/?PROMPT(A  : )/,
bv/?PROMPT(B  : )/,
y1/?PROMPT(Starting Y       : )/,
y2/?PROMPT(Ending Y         : )/)
({
  ?MESSAGE()
  ?MESSAGE(Transform will be Y = a X + b)
  ?MESSAGE(Please enter a and b value.)
  ?MESSAGE(Note : Parameter a should not be 0.)
  ?MESSAGE()
  ?T(a)_av
  ?T(b)_bv
  title1_"Transform of Standard Normal Variable with Y = a X + b"
  title2_encode("A :",?T(a)," B :",?T(b))
  subject_c(title1, title2)
  ?MESSAGE()
  ?MESSAGE(Please enter input about y.)
  ?MESSAGE()
  ?T(y)_seq(y1,y2, len=51)
  ?T(x)_(?T(y)-?T(b))/?T(a)
#
# The pdf for transformed variable can be computed as
# g(y) = f( (y-b)/a ) * 1/a
#
  ?T(t)_dnorm((?T(y)-?T(b))/?T(a))/?T(a)
#
# The pdf for original variable, which is standard normal
# distribtion.
#
  ?T(z)_dnorm(?T(x))
#
# Call macro to plot y = a X + b
#
  ?lineartran1(subject,?T(y),?T(x))
#
# Call macro to plot pdf transformation
#
  ?lineartran2(subject,?T(y),?T(x),?T(t),?T(z))
  rm(?T(x),?T(y),?T(t),?T(z),?T(a),?T(b), subject, title1, title2)
})
END
```

```
MACRO lineartran1(subj, y, x)
({
  par(mfrow=c(1,2), oma=c(3,0,2,0))
  plot(x,y, type="l", xlab="X", ylab="Y", err=-1)
  plot(y,x, type="l", xlab="Y", ylab="X", err=-1)
  mtext(side=3, line=0, outer=T,
    encode(subj[1]))
  mtext(side=1, line=0, outer=T,
    encode(subj[2]))
})
END
```

```
MACRO lineartran2(subj,y,x,t,z)
({
  par(mfrow=c(1,2), oma=c(3,3,3,0))
  my_max(t,z)
  plot(y,t, xlab="Y", ylab="", type="l", ylim=c(0,my),
       err=-1)
  plot(x,z, xlab="X", ylab="", type="l", ylim=c(0,my),
       err=-1)
  mtext(side=3, line=1, outer=T,
    encode(subj[1]))
  mtext(side=2, line=1, outer=T, "PDF")
  mtext(side=1, line=1, outer=T,
    encode(subj[2]))
  rm(my)
})
END
```

```
MACRO trnorm(
mv/?PROMPT(Mean for X          : )/,
sv/?PROMPT(Std Dev for X       : )/)
({
  ?T(m)_mv
  ?T(s)_sv
  title1_"Transform of Normal Variable"
  title2_encode("Mu :",?T(m)," S :",?T(s))
  subject_c(title1, title2)
  para_c(?T(m),?T(s))
  ?MESSAGE()
  ?MESSAGE(Normal variable is defined only in X > 0 for transform 2,)
  ?MESSAGE(and X >= 0 for transform 4.)
  ?MESSAGE(Please enter proper range of Y if you choose one of them.)
  ?MESSAGE()
  ?MESSAGE(Type of Transformation : )
  ?MESSAGE()
  item_c("y = x^2",
         "y = log (x)",
         "y = exp (x)",
         "y = sqrt (x)")
  action_c("?tr1nt(norm,para,subject)",
           "?tr2(norm,para,subject)",
           "?tr3(norm,para,subject)",
           "?tr4(norm,para,subject)")
   menu(item,action)
   rm(?T(m),?T(s),title1,title2,item,action)
})
END
```

```
MACRO trunif(
lb/?PROMPT(Lower Bound     : )/,
ub/?PROMPT(Upper Bound     : )/)
({
  lower_lb
  upper_ub
  para_c(lower,upper)
  title1_"Transform of Uniform Variable"
  title2_encode("LB :",lower," UB :",upper)
  subject_c(title1, title2)
  ?MESSAGE()
  ?MESSAGE(Type of Transformation : )
  ?MESSAGE()
  item_c("y = x^2",
         "y = log (x)",
         "y = exp (x)",
         "y = sqrt (x)")
  action_c("?unifins1( (max(0,lower,-upper))^2,
              (max(upper,-lower))^2 ) ;
           ?tr1u(subject,para,(max(0,lower,-upper))^2,
                 (max(upper,-lower))^2)",

           "?unifins2( log(max(lower,0.00001)), log(upper)) ;
           ?tr2(unif,para,subject)",

           "?unifins2(exp(lower), exp(upper)) ;
           ?tr3(unif,para,subject)",

           "?unifins2(sqrt(max(lower,0)), sqrt(upper)) ;
           ?tr4(unif,para,subject)")
  menu( item, action)
  rm(title1, title2, action, item)
})
END
```

```
MACRO trlnorm(
mv/?PROMPT(Mean for X          : )/,
sv/?PROMPT(Std Dev for X       : )/)
({
  ?T(m)_mv
  ?T(s)_sv
  title1_"Transform of Lognormal Variable"
  title2_encode("Mu :",?T(m)," S :",?T(s))
  subject_c(title1, title2)
  para_c(?T(m),?T(s))
  ?MESSAGE()
  ?MESSAGE(  Lognormal variables are defined only X > 0.)
  ?MESSAGE(  Please enter proper range of Y for desired result.)
  ?MESSAGE()
  ?MESSAGE(Type of Transformation : )
  ?MESSAGE()
  item_c("y = x^2",
         "y = log (x)",
         "y = exp (x)",
         "y = sqrt (x)")
  action_c("?tr1(lnorm,para,subject)",
           "?tr2(lnorm,para,subject)",
           "?tr3(lnorm,para,subject)",
           "?tr4(lnorm,para,subject)")
   menu( item,action)
   rm(?T(m),?T(s),title1,title2,item,action)
})
END
```

```
MACRO trexp(
lm/?PROMPT(Lambda for X       : )/)
({
  ?T(1)_lm
  title1_"Transform of Exponential Variable"
  title2_encode("Lambda :",?T(1))
  subject_c(title1, title2)
  para_?T(1)
  ?MESSAGE()
  ?MESSAGE(  Exponential variables are defined only X > 0.)
  ?MESSAGE(  Please enter proper range of Y for desired result.)
  ?MESSAGE()
  ?MESSAGE(Type of Transformation : )
  ?MESSAGE()
   item_c("y = x^2",
          "y = log (x)",
          "y = exp (x)",
          "y = sqrt (x)")
  action_c("?ttr1(ex1,para,subject)",
           "?ttr2(ex1,para,subject)",
           "?ttr3(ex1,para,subject)",
           "?ttr4(ex1,para,subject)")
   menu( item,action)
   rm(?T(1),title1,title2, item,action)
})
END
```

```
MACRO trgamma(
sh/?PROMPT(Parameter  alpha  : )/)
({
  ?T(n)_sh
  title1_"Transform of  Std. Gamma Variable"
  title2_encode("Alpha :",?T(n))
  subject_c(title1, title2)
  para_?T(n)
  ?MESSAGE()
  ?MESSAGE(  Gamma variables are defined only X > 0.)
  ?MESSAGE(  Please enter proper range of Y for desired result.)
  ?MESSAGE()
  ?MESSAGE(Type of Transformation : )
  ?MESSAGE()
  item_c("y = x^2",
         "y = log (x)",
         "y = exp (x)",
         "y = sqrt (x)")
  action_c("?tr1(gamma,para,subject)",
           "?tr2(gamma,para,subject)",
           "?tr3(gamma,para,subject)",
           "?tr4(gamma,para,subject)")
   menu( item,action)
   rm(?T(n),title1,title2,item,action)
})
END
```

```
MACRO trf(
nd/?PROMPT(DF for Numerater   : )/,
dd/?PROMPT(DF for Denominator : )/)
({
  ?T(n)_nd
  ?T(d)_dd
  title1_"Transform of F Variable"
  title2_encode("DF 1 :",?T(n)," DF 2 :",?T(d))
  subject_c(title1, title2)
  para_c(?T(n),?T(d))
  ?MESSAGE()
  ?MESSAGE(  F variables are defined only X > 0.)
  ?MESSAGE(  Please enter proper range of Y for desired result.)
  ?MESSAGE()
  ?MESSAGE(Type of Transformation : )
  ?MESSAGE()
  item_c("y = x^2",
         "y = log (x)",
         "y = exp (x)",
         "y = sqrt (x)",
         "y = a X + b")
  action_c("?tr1(f,para,subject)",
           "?tr2(f,para,subject)",
           "?tr3(f,para,subject)",
           "?tr4(f,para,subject)",
           "?trfline(para)")
   menu(item,action)
   rm(?T(n),?T(d),title1,title2,item,action)
})
END
```

```
MACRO trfline(dof,
av/?PROMPT(A  : )/,
bv/?PROMPT(B  : )/,
y1/?PROMPT(Starting Y      : )/,
y2/?PROMPT(Ending Y        : )/)
({
  ?MESSAGE()
  ?MESSAGE(Transform will be Y = a X + b)
  ?MESSAGE(Please enter a and b value.)
  ?MESSAGE(Note : Parameter a should not be 0.)
  ?MESSAGE()
  ?T(a)_av
  ?T(b)_bv
  title1_"Transform of Chisquare Variable with Y = a X + b"
  title2_encode("D.F. 1 :",dof[1]," D.F. 2 :",dof[2],"   A :",
              ?T(a)," B :",?T(b))
  subject_c(title1, title2)
  ?MESSAGE()
  ?MESSAGE(Please enter input about y.)
  ?MESSAGE()
  ?T(y)_seq(y1,y2, len=51)
  ?T(x)_(?T(y)-?T(b))/?T(a)
#
# To avoid floating in S computation, zero f variable
#  is eliminated.
#
  ?T(x)_?T(x)[?T(x)!=0]
  ?T(y)_?T(y)[?T(y)!=?T(b)]
#
# The pdf for transformed variable can be computed as
# g(y) = f( (y-b)/a ) * 1/a
#
  ?T(t)_df(((?T(y)-?T(b))/?T(a)), dof[1], dof[2])/?T(a)
#
# The pdf for original variable, which is F distribtion.
#
  ?T(z)_df(?T(x),dof[1],dof[2])
#
# Call macro to plot y = a X + b
#
  ?lineartran1(subject,?T(y),?T(x))
#
# Call macro to plot pdf transformation
#
  ?lineartran2(subject,?T(y),?T(x),?T(t),?T(z))
  rm(?T(x),?T(y),?T(t),?T(z),?T(a),?T(b), subject, title1, title2)
})
END
```

```
MACRO trt(
df/?PROMPT(Degree of Freedom  : )/)
({
  ?T(f)_df
  title1_"Transform of T Variable"
  title2_encode("Degree of Freedom :",?T(f))
  subject_c(title1, title2)
  para_?T(f)
  ?MESSAGE()
  ?MESSAGE(T variable is defined only in X > 0 for transform 2,)
  ?MESSAGE( and X >= 0 for transform 4.)
  ?MESSAGE(Please enter proper range of Y if you choose one of them.)
  ?MESSAGE()
  ?MESSAGE(Type of Transformation : )
  ?MESSAGE()
   item_c("y = x^2",
          "y = log (x)",
          "y = exp (x)",
          "y = sqrt (x)")
  action_c("?tr1nt(t,para,subject)",
            "?tr2(t,para,subject)",
            "?tr3(t,para,subject)",
            "?tr4(t,para,subject)")
   menu( item,action)
   rm(?T(f),title1,title2, item,action)
})
END
```

```
MACRO trweib(
sh/?PROMPT(Shape parameter    : )/,
sc/?PROMPT(Scale parameter    : )/)
({
  ?T(n)_sh
  ?T(d)_sc
  title1_"Transform of Weibull Variable"
  title2_encode("Shape :",?T(n)," Scale :",?T(d))
  subject_c(title1, title2)
  para_c(?T(n),?T(d))
  ?MESSAGE()
  ?MESSAGE(  Weibull variables are defined only X > 0.)
  ?MESSAGE(  Please enter proper range of Y for desired result.)
  ?MESSAGE()
  ?MESSAGE(Type of Transformation : )
  ?MESSAGE()
  item_c("y = x^2",
         "y = log (x)",
         "y = exp (x)",
         "y = sqrt (x)")
  action_c("?ttr1(weib,para,subject)",
           "?ttr2(weib,para,subject)",
           "?ttr3(weib,para,subject)",
           "?ttr4(weib,para,subject)")
   menu( item,action)
   rm(?T(n),?T(d),title1,title2, item,action)
})
END
```

```
MACRO trchisq(
df/?PROMPT(Degree of Freedom  : )/)
({
  ?T(d)_df
  title1_"Transform of Chisquare distribution Variable"
  title2_encode("DF :",?T(d))
  subject_c(title1, title2)
  para_?T(d)
  ?MESSAGE()
  ?MESSAGE(  Chisquare variables are defined only X > 0.)
  ?MESSAGE(  Please enter proper range of Y for desired result.)
  ?MESSAGE()
  ?MESSAGE(Type of Transformation : )
  ?MESSAGE()
  item_c("y = x^2",
         "y = log (x)",
         "y = exp (x)",
         "y = sqrt (x)",
         "y = a X + b")
  action_c("?tr1(chisq,para,subject)",
           "?tr2(chisq,para,subject)",
           "?tr3(chisq,para,subject)",
           "?tr4(chisq,para,subject)",
           "?trchline(para)")
   menu( item,action)
   rm(?T(d),title1,title2, item,action)
})
END
```

```
MACRO trchline(df,
av/?PROMPT(A  : )/,
bv/?PROMPT(B  : )/,
y1/?PROMPT(Starting Y      : )/,
y2/?PROMPT(Ending Y        : )/)
({
  ?MESSAGE()
  ?MESSAGE(Transform will be Y = a X + b)
  ?MESSAGE(Please enter a and b value.)
  ?MESSAGE(Note :  a should not be 0.)
  ?MESSAGE()
  ?T(a)_av
  ?T(b)_bv
  title1_"Transform of Chisquare Variable with Y = a X + b"
  title2_encode("D.F. :",df,"  A :",?T(a)," B :",?T(b))
  subject_c(title1, title2)
  ?MESSAGE()
  ?MESSAGE(Please enter input about y.)
  ?MESSAGE()
  ?T(y)_seq(y1,y2, len=51)
  ?T(x)_(?T(y)-?T(b))/?T(a)
#
# To avoid floating in S computation, zero chisquare variable
#  is eliminated.
#
  ?T(x)_?T(x)[?T(x)!=0]
  ?T(y)_?T(y)[?T(y)!=?T(b)]
#
# The pdf for transformed variable can be computed as
# g(y) = f( (y-b)/a ) * 1/a
#
  ?T(t)_dchisq(((?T(y)-?T(b))/?T(a)),df)/?T(a)
#
# The pdf for original variable, which is Chisquare distribtion.
#
  ?T(z)_dchisq(?T(x),df)
#
# Call macro to plot y = a X + b
#
  ?lineartran1(subject,?T(y),?T(x))
#
# Call macro to plot pdf transformation
#
  ?lineartran2(subject,?T(y),?T(x),?T(t),?T(z))
  rm(?T(x),?T(y),?T(t),?T(z),?T(a),?T(b), subject, title1, title2)
})
END
```

```
MACRO tr1(dis,para,subject,
sp/?PROMPT(Starting Pt for Y  : )/,
ep/?PROMPT(Ending Pt for Y    : )/,
dv/?PROMPT(Increment for Y    : )/)
#
# This Macro is to compute the hight of transformed variable,
# which is the estimated PDF for the transformed variable,
# when the transformation is y=x^2 and it has pdf and cdf
# function in S.
# And such distribution as normal or t can not use this macro,
# instead use other one named tr1nt, since not only positive
# X value but also negative X value should be considered.
#
({
  ?T(b)_sp
  ?T(e)_ep
  ?T(d)_dv
  np_len(para)
  ?T(y)_seq(?T(b),?T(e),?T(d))  # Y  : Y range [b e]
  ?T(x)_sqrt(?T(y))             # X  : Transformation
  ?T(n)_len(?T(y))              # N  : length of Y
  ?T(l)_4*?T(n)-3
  maxy_max(?T(y))
  ?T(py)_seq(?T(b),maxy,len=?T(l))  # PY : Y with small interval
  ?T(yy)_seq(?T(b),maxy,len=51)     # YY : Y for Plot
  ?T(px)_sqrt(?T(py))               # PX : X for PY
  ?T(xx)_sqrt(?T(yy))               # XX : X for YY
#
# If given distribution has function and 1 parameter
#
   if(np == 1) {
     ?T(pd)_?(d)dis(?T(x),para[1])   # PD : PDF for X
     ?T(tt)_?(d)dis(?T(xx),para[1])   # TT : PDF for XX
     ?T(td)_?(d)dis(?T(px),para[1])  # TD : PDF for PX
     ?T(cd)_?(p)dis(?T(x),para[1])   # CD : CDF for X
     } else {
#
# If given distribution has function and 2 parameter
#
     ?T(pd)_?(d)dis(?T(x),para[1],para[2])   # PD : PDF for X
     ?T(tt)_?(d)dis(?T(xx),para[1],para[2])  # TT : PDF for XX
     ?T(td)_?(d)dis(?T(px),para[1],para[2])  # TD : PDF for PX
     ?T(cd)_?(p)dis(?T(x),para[1],para[2])   # CD : CDF for X
     }
#
# Area between Y[i] and Y[i+1]
#
  ?T(ay)_?T(cd)[2:?T(n)]-?T(cd)[1:(?T(n)-1)]
```

```
#
# Hight(estimated PDF for Y) can be computed by Area/Increment
#
  subject_c(subject,"Y = (X) ^2")
  ?T(hi)_?T(ay)/?T(d)
#
# Call macro trplot with following argument
#
  ?trplot(?T(x),?T(y),?T(pd),?T(hi),?T(px),?T(py),
          ?T(d),?T(td),?T(n),subject,?T(xx),?T(yy),?T(tt))
rm(?T(b),?T(e),?T(cd),?T(ay),?T(l),para,np,maxy)
})
END
```

```
MACRO tr2(dis,para,subject,
sp/?PROMPT(Starting Pt for Y  : )/,
ep/?PROMPT(Ending Pt for Y    : )/,
dv/?PROMPT(Increment for Y    : )/)
#
# This Macro is to compute the hight of transformed variable,
# which is the estimated PDF for the transformed variable,
# when the transformation is y=log(x) and it has pdf and cdf
# function in S.
#
({
  ?T(b)_sp
  ?T(e)_ep
  ?T(d)_dv
  np_len(para)
  ?T(y)_seq(?T(b),?T(e),?T(d))  # Y  : Y range [b e]
  ?T(x)_exp(?T(y))              # X  : Transformation
  ?T(n)_len(?T(y))              # N  : length of Y
  ?T(l)_4*?T(n)-3
  maxy_max(?T(y))
  ?T(py)_seq(?T(b),maxy,len=?T(l))  # PY : Y with small interval
  ?T(yy)_seq(?T(b),maxy,len=51)     # YY : Y for Plot
  ?T(px)_exp(?T(py))                # PX : X for PY
  ?T(xx)_exp(?T(yy))                # XX : X for YY
#
# If given distribution has function and 1 parameter
#
   if(np == 1) {
     ?T(pd)_?(d)dis(?T(x),para[1])    # PD : PDF for X
     ?T(tt)_?(d)dis(?T(xx),para[1])   # TT : PDF for XX
     ?T(td)_?(d)dis(?T(px),para[1])   # TD : PDF for PX
     ?T(cd)_?(p)dis(?T(x),para[1])    # CD : CDF for X
     } else {
#
# If given distribution has function and 2 parameter
#
     ?T(pd)_?(d)dis(?T(x),para[1],para[2])    # PD : PDF for X
     ?T(tt)_?(d)dis(?T(xx),para[1],para[2])   # TT : PDF for XX
     ?T(td)_?(d)dis(?T(px),para[1],para[2])   # TD : PDF for PX
     ?T(cd)_?(p)dis(?T(x),para[1],para[2])    # CD : CDF for X
     }
#
# Area between Y[i] and Y[i+1]
#
  ?T(ay)_?T(cd)[2:?T(n)]-?T(cd)[1:(?T(n)-1)]
#
# Hight(estimated PDF for Y) can be computed by Area/Increment
#
  subject_c(subject,"Y = Log (X)")
  ?T(hi)_?T(ay)/?T(d)
```

```
#
# Call macro trplot with following argument
#
  ?trplot(?T(x),?T(y),?T(pd),?T(hi),?T(px),?T(py),
          ?T(d),?T(td),?T(n),subject,?T(xx),?T(yy),?T(tt))
rm(?T(b),?T(e),?T(cd),?T(ay),?T(l),para,np,maxy)
})
END
```

```
MACRO tr3( dis,para,subject,
sp/?PROMPT(Starting Pt for Y  : )/,
ep/?PROMPT(Ending Pt for Y    : )/,
dv/?PROMPT(Increment for Y    : )/)
#
# This Macro is to compute the hight of transformed variable,
# which is the estimated PDF for the transformed variable,
# when the transformation is y=exp(x) and it has pdf and cdf
# function in S.
#
({
  ?T(b)_sp
  ?T(e)_ep
  ?T(d)_dv
  np_len(para)
  ?T(y)_seq(?T(b),?T(e),?T(d))  # Y  : Y range [b e]
  ?T(x)_log(?T(y))              # X  : Transformation
  ?T(n)_len(?T(y))              # N  : length of Y
  ?T(l)_4*?T(n)-3
  maxy_max(?T(y))
  ?T(py)_seq(?T(b),maxy,len=?T(l))  # PY : Y with small interval
  ?T(yy)_seq(?T(b),maxy,len=51)     # YY : Y for Plot
  ?T(px)_log(?T(py))                # PX : X for PY
  ?T(xx)_log(?T(yy))                # XX : X for YY
#
# If given distribution has function and 1 parameter
#
   if(np == 1) {
     ?T(pd)_?(d)dis(?T(x),para[1])    # PD : PDF for X
     ?T(tt)_?(d)dis(?T(xx),para[1])   # TT : PDF for XX
     ?T(td)_?(d)dis(?T(px),para[1])   # TD : PDF for PX
     ?T(cd)_?(p)dis(?T(x),para[1])    # CD : CDF for X
     } else {
#
# If given distribution has function and 2 parameter
#
     ?T(pd)_?(d)dis(?T(x),para[1],para[2])    # PD : PDF for X
     ?T(tt)_?(d)dis(?T(xx),para[1],para[2])   # TT : PDF for XX
     ?T(td)_?(d)dis(?T(px),para[1],para[2])   # TD : PDF for PX
     ?T(cd)_?(p)dis(?T(x),para[1],para[2])    # CD : CDF for X
     }
#
# Area between Y[i] and Y[i+1]
#
  ?T(ay)_?T(cd)[2:?T(n)]-?T(cd)[1:(?T(n)-1)]
#
# Hight(estimated PDF for Y) can be computed by Area/Increment
#
  subject_c(subject,"Y = Exp (X)")
  ?T(hi)_?T(ay)/?T(d)
```

```
#
# Call macro trplot with following argument
#
  ?trplot(?T(x),?T(y),?T(pd),?T(hi),?T(px),?T(py),
          ?T(d),?T(td),?T(n),subject,?T(xx),?T(yy),?T(tt))
rm(?T(b),?T(e),?T(cd),?T(ay),?T(l),para,np,maxy)
})
END
```

```
MACRO tr4( dis,para,subject,
sp/?PROMPT(Starting Pt for Y  : )/,
ep/?PROMPT(Ending Pt for Y    : )/,
dv/?PROMPT(Increment for Y    : )/)
#
# This Macro is to compute the hight of transformed variable,
# which is the estimated PDF for the transformed variable,
# when the transformation is y=sqrt(x) and it has pdf and cdf
# function in S.
#
({
  ?T(b)_sp
  ?T(e)_ep
  ?T(d)_dv
  np_len(para)
  ?T(y)_seq(?T(b),?T(e),?T(d))  # Y  : Y range [b e]
  ?T(x)_(?T(y))^2               # X  : Transformation
  ?T(n)_len(?T(y))              # N  : length of Y
  ?T(l)_4*?T(n)-3
  maxy_max(?T(y))
  ?T(py)_seq(?T(b),maxy,len=?T(l))  # PY : Y with small interval
  ?T(yy)_seq(?T(b),maxy,len=51)     # YY : Y for Plot
  ?T(px)_(?T(py))^2                 # PX : X for PY
  ?T(xx)_(?T(yy))^2                 # XX : X for YY
#
# If given distribution has function and 1 parameter
#
   if(np == 1) {
     ?T(pd)_?(d)dis(?T(x),para[1])   # PD : PDF for X
     ?T(tt)_?(d)dis(?T(xx),para[1])  # TT : PDF for XX
     ?T(td)_?(d)dis(?T(px),para[1])  # TD : PDF for PX
     ?T(cd)_?(p)dis(?T(x),para[1])   # CD : CDF for X
     } else {
#
# If given distribution has function and 2 parameter
#
     ?T(pd)_?(d)dis(?T(x),para[1],para[2])   # PD : PDF for X
     ?T(tt)_?(d)dis(?T(xx),para[1],para[2])  # TT : PDF for XX
     ?T(td)_?(d)dis(?T(px),para[1],para[2])  # TD : PDF for PX
     ?T(cd)_?(p)dis(?T(x),para[1],para[2])   # CD : CDF for X
     }
#
# Area between Y[i] and Y[i+1]
#
  ?T(ay)_?T(cd)[2:?T(n)]-?T(cd)[1:(?T(n)-1)]
#
# Hight(estimated PDF for Y) can be computed by Area/Increment
#
  subject_c(subject,"Y = Sqrt (X)")
  ?T(hi)_?T(ay)/?T(d)
```

```
#
# Call macro trplot with following argument
#
  ?trplot(?T(x),?T(y),?T(pd),?T(hi),?T(px),?T(py),
          ?T(d),?T(td),?T(n),subject,?T(xx),?T(yy),?T(tt))
rm(?T(b),?T(e),?T(cd),?T(ay),?T(l),para,np,maxy)
})
END
```

```
MACRO ttr1( dis,para,subject,
sp/?PROMPT(Starting Pt for Y  : )/,
ep/?PROMPT(Ending Pt for Y    : )/,
dv/?PROMPT(Increment for Y    : )/)
#
# This Macro is to compute the hight of transformed variable,
# which is the estimated PDF for the transformed variable,
# when the transformation is y=x^2 and it has pdf and cdf
# macro not function in S.
#
({
  ?T(b)_sp
  ?T(e)_ep
  ?T(d)_dv
  np_len(para)
  ?T(y)_seq(?T(b),?T(e),?T(d))  # Y  : Y range [b e]
  ?T(x)_sqrt(?T(y))             # X  : Transformation
  ?T(n)_len(?T(y))              # N  : length of Y
  ?T(l)_4*?T(n)-3
  maxy_max(?T(y))
  ?T(py)_seq(?T(b),maxy,len=?T(l))  # PY : Y with small interval
  ?T(yy)_seq(?T(b),maxy,len=51)     # YY : Y for Plot
  ?T(px)_sqrt(?T(py))               # PX : X for PY
  ?T(xx)_sqrt(?T(yy))               # XX : X for YY
  ?T(pd)_?(?d)dis(?T(x),para[1],para[np])   # PD : PDF for X
  ?T(tt)_?(?d)dis(?T(xx),para[1],para[np])  # TT : PDF for XX
  ?T(td)_?(?d)dis(?T(px),para[1],para[np])  # TD : PDF for PX
  ?T(cd)_?(?p)dis(?T(x),para[1],para[np])   # CD : CDF for X
#
# Area between Y[i] and Y[i+1]
#
  ?T(ay)_?T(cd)[2:?T(n)]-?T(cd)[1:(?T(n)-1)]
#
# Hight(estimated PDF for Y) can be computed by Area/Increment
#
  subject_c(subject,"Y = (X) ^2")
  ?T(hi)_?T(ay)/?T(d)
#
# Call macro trplot with following argument
#
  ?trplot(?T(x),?T(y),?T(pd),?T(hi),?T(px),?T(py),
          ?T(d),?T(td),?T(n),subject,?T(xx),?T(yy),?T(tt))
rm(?T(b),?T(e),?T(cd),?T(ay),?T(l),para,np,maxy)
})
END
```

```
MACRO ttr2( dis,para,subject,
sp/?PROMPT(Starting Pt for Y  : )/,
ep/?PROMPT(Ending Pt for Y    : )/,
dv/?PROMPT(Increment for Y    : )/)
#
# This Macro is to compute the hight of transformed variable,
# which is the estimated PDF for the transformed variable,
# when the transformation is y=log(x) and it has pdf and cdf
# macro not function in S.
#
({
  ?T(b)_sp
  ?T(e)_ep
  ?T(d)_dv
  np_len(para)
  ?T(y)_seq(?T(b),?T(e),?T(d))  # Y  : Y range [b e]
  ?T(x)_exp(?T(y))              # X  : Transformation
  ?T(n)_len(?T(y))              # N  : length of Y
  ?T(l)_4*?T(n)-3
  maxy_max(?T(y))
  ?T(py)_seq(?T(b),maxy,len=?T(l))  # PY : Y with small interval
  ?T(yy)_seq(?T(b),maxy,len=51)     # YY : Y for Plot
  ?T(px)_exp(?T(py))                # PX : X for PY
  ?T(xx)_exp(?T(yy))                # XX : X for YY
  ?T(pd)_?(?d)dis(?T(x),para[1],para[np])   # PD : PDF for X
  ?T(tt)_?(?d)dis(?T(xx),para[1],para[np])  # TT : PDF for XX
  ?T(td)_?(?d)dis(?T(px),para[1],para[np])  # TD : PDF for PX
  ?T(cd)_?(?p)dis(?T(x),para[1],para[np])   # CD : CDF for X
#
# Area between Y[i] and Y[i+1]
#
  ?T(ay)_?T(cd)[2:?T(n)]-?T(cd)[1:(?T(n)-1)]
#
# Hight(estimated PDF for Y) can be computed by Area/Increment
#
  subject_c(subject,"Y = Log (X)")
  ?T(hi)_?T(ay)/?T(d)
#
# Call macro trplot with following argument
#
  ?trplot(?T(x),?T(y),?T(pd),?T(hi),?T(px),?T(py),
          ?T(d),?T(td),?T(n),subject,?T(xx),?T(yy),?T(tt))
rm(?T(b),?T(e),?T(cd),?T(ay),?T(l),para,np,maxy)
})
END
```

```
MACRO ttr3( dis,para,subject,
sp/?PROMPT(Starting Pt for Y  : )/,
ep/?PROMPT(Ending Pt for Y    : )/,
dv/?PROMPT(Increment for Y    : )/)
#
# This Macro is to compute the hight of transformed variable,
# which is the estimated PDF for the transformed variable,
# when the transformation is y=exp(x) and it has pdf and cdf
# macro not function in S.
#
({
  ?T(b)_sp
  ?T(e)_ep
  ?T(d)_dv
  np_len(para)
  ?T(y)_seq(?T(b),?T(e),?T(d))  # Y   : Y range [b e]
  ?T(x)_log(?T(y))              # X   : Transformation
  ?T(n)_len(?T(y))              # N   : length of Y
  ?T(l)_4*?T(n)-3
  maxy_max(?T(y))
  ?T(py)_seq(?T(b),maxy,len=?T(l))  # PY : Y with small interval
  ?T(yy)_seq(?T(b),maxy,len=51)     # PY : Y for Plot
  ?T(px)_log(?T(py))                # PX : X for PY
  ?T(xx)_log(?T(yy))                # PX : X for YY
  ?T(pd)_?(?d)dis(?T(x),para[1],para[np])    # PD : PDF for X
  ?T(tt)_?(?d)dis(?T(xx),para[1],para[np])   # TT : PDF for XX
  ?T(td)_?(?d)dis(?T(px),para[1],para[np])   # TD : PDF for PX
  ?T(cd)_?(?p)dis(?T(x),para[1],para[np])    # CD : CDF for X
#
# Area between Y[i] and Y[i+1]
#
  ?T(ay)_?T(cd)[2:?T(n)]-?T(cd)[1:(?T(n)-1)]
#
# Hight(estimated PDF for Y) can be computed by Area/Increment
#
  subject_c(subject,"Y = Exp (X)")
  ?T(hi)_?T(ay)/?T(d)
#
# Call macro trplot with following argument
#
  ?trplot(?T(x),?T(y),?T(pd),?T(hi),?T(px),?T(py),
          ?T(d),?T(td),?T(n),subject,?T(xx),?T(yy),?T(tt))
rm(?T(b),?T(e),?T(cd),?T(ay),?T(l),para,np,maxy)
})
END
```

```
MACRO ttr4( dis,para,subject,
sp/?PROMPT(Starting Pt for Y  : )/,
ep/?PROMPT(Ending Pt for Y    : )/,
dv/?PROMPT(Increment for Y    : )/)
#
# This Macro is to compute the hight of transformed variable,
# which is the estimated PDF for the transformed variable,
# when the transformation is y=sqrt(x) and it has pdf and cdf
# macro not function in S.
#
({
  ?T(b)_sp
  ?T(e)_ep
  ?T(d)_dv
  np_len(para)
  ?T(y)_seq(?T(b),?T(e),?T(d))  # Y  : Y range [b e]
  ?T(x)_(?T(y))^2               # X  : Transformation
  ?T(n)_len(?T(y))              # N  : length of Y
  ?T(l)_4*?T(n)-3
  maxy_max(?T(y))
  ?T(py)_seq(?T(b),maxy,len=?T(l))  # PY : Y with small interval
  ?T(yy)_seq(?T(b),maxy,len=51)     # PY : Y for Plot
  ?T(px)_(?T(py))^2                 # PX : X for PY
  ?T(xx)_(?T(yy))^2                 # PX : X for YY
  ?T(pd)_?(?d)dis(?T(x),para[1],para[np])    # PD : PDF for X
  ?T(tt)_?(?d)dis(?T(xx),para[1],para[np])   # TT : PDF for XX
  ?T(td)_?(?d)dis(?T(px),para[1],para[np])   # TD : PDF for PX
  ?T(cd)_?(?p)dis(?T(x),para[1],para[np])    # CD : CDF for X
#
# Area between Y[i] and Y[i+1]
#
  ?T(ay)_?T(cd)[2:?T(n)]-?T(cd)[1:(?T(n)-1)]
#
# Hight(estimated PDF for Y) can be computed by Area/Increment
#
  subject_c(subject,"Y = Sqrt (X)")
  ?T(hi)_?T(ay)/?T(d)
#
# Call macro trplot with following argument
#
  ?trplot(?T(x),?T(y),?T(pd),?T(hi),?T(px),?T(py),
          ?T(d),?T(td),?T(n),subject,?T(xx),?T(yy),?T(tt))
rm(?T(b),?T(e),?T(cd),?T(ay),?T(l),para,np,maxy)
})
END
```

```
MACRO trint(dis,para,subject,
sp/?PROMPT(Starting Pt for Y  : )/,
ep/?PROMPT(Ending Pt for Y    : )/,
dv/?PROMPT(Increment for Y    : )/)
#
# This Macro is to compute the hight of transformed variable,
# which is the estimated PDF for the transformed variable,
# when the transformation is y=x^2, in which x is defined
# from negative infinite to positive infinite such as normal
# or t distribution, and it has pdf and cdf function in S.
#
({
  ?T(b)_sp
  ?T(e)_ep
  ?T(d)_dv
  np_len(para)
  ?T(y)_seq(?T(b),?T(e),?T(d))  # Y  : Y range [b e]
  maxy_max(?T(y))
  ?T(n)_len(?T(y))                  # N  : length of Y
  ?T(l)_4*?T(n)-3
  ?T(yy)_seq(?T(b),maxy,len=?T(l))
  ?T(x)_sqrt(?T(y))             # X  : Transformation
#
# For defined Y, two X values(same value but different sign)
#
  ?T(x)_c(-rev(?T(x)),?T(x))        # X  : ([-e -b] [b e])
                                    #      with given interval
  maxx_sqrt(maxy)
  minx_sqrt(?T(b))
  ?T(xx)_sqrt(?T(yy))
  ?T(xx)_c(-rev(?T(xx)),?T(xx))     # XX : Same as X but with small int
  ?T(rx)_seq(0,minx,len=15)
  ?T(rx)_c(-rev(?T(rx)),?T(rx))     # RX : [-b b] with small int
  ?T(px)_seq(-maxx,maxx,len=51)     # PX : [-e e] with small int
  ?T(py)_?T(px)^2                   # PY : Y for given PX
#
# When 2 parameters are required ;
#
   if(np==2) {
  ?T(pd)_?(d)dis(?T(x),para[1],para[2])   # PD : PDF for X
  ?T(td)_?(d)dis(?T(px),para[1],para[2])  # TD : PDF for PX
  ?T(dd)_?(d)dis(?T(xx),para[1],para[2])  # DD : PDF for XX
  ?T(rd)_?(d)dis(?T(rx),para[1],para[2])  # RD : PDF for RX
  ?T(cd)_?(p)dis(?T(x),para[1],para[2])   # CD : CDF for X
  }  else {
```

```
#
# When 1 parameter is required ;
#
  ?T(pd)_?(d)dis(?T(x),para)            # PD : PDF for X
  ?T(td)_?(d)dis(?T(px),para)           # TD : PDF for PX
  ?T(dd)_?(d)dis(?T(xx),para)           # DD : PDF for XX
  ?T(rd)_?(d)dis(?T(rx),para)           # RD : PDF for RX
  ?T(cd)_?(p)dis(?T(x),para)            # CD : CDF for X
  }
#
# Area between x[i] and x[i+1], which is sum of
# two areas.
#
  ?T(ay1)_?T(cd)[1:?T(n)]
  ?T(ay2)_?T(cd)[(?T(n)+1):(2*?T(n))]
  ?T(ay1)_?T(ay1)[2:?T(n)]-?T(ay1)[1:(?T(n)-1)]
  ?T(ay2)_?T(ay2)[2:?T(n)]-?T(ay2)[1:(?T(n)-1)]
  ?T(ay1)_rev(?T(ay1))
  ?T(ay)_?T(ay1)+?T(ay2)
#
# Hight for given Y (estimated pdf) is obtained from
# area/increment.
#
  ?T(hi)_?T(ay)/?T(d)
  subject_c(subject,"Y = X ^2")
#
# Call macro trtplot with following arguments.
#
  ?trtplot(?T(x),?T(y),?T(pd),?T(hi),?T(px),?T(py),?T(d),
           ?T(td),?T(n),?T(xx),?T(dd),?T(rx),?T(rd),subject)
rm(?T(b),?T(e),?T(ay),?T(dy),?T(l),?T(ay1),
   ?T(ay2),?T(cd),dis,prar,maxy,maxx,minx)
})
END
```

```
MACRO trplot(x,y,pd,hi,px,py,dd,td,nn,subject,tx,ty,tt)
({
# X  : X [b e]
# Y  : Transformed Y
# PD : PDF for X
# HI : Hight(estimated CDF) for Y
# PX : X with small interval
# PY : Y for PX
# DD : Given interval for X
# TD : PDF for PX
# NN : length of Y
# Subject : Related title
# TX : X for Plot
# TY : Y for Plot
# TT : PDF for XX
#
  par(mfrow=c(1,2),oma=c(5,0,5,0))
#
# Plot X vs. Y
#
  plot(tx,ty,type="l",xlab="X",ylab="Y")
#
# Plot Y vs. X
#
  plot(ty,tx,type="l",xlab="Y",ylab="X")
  mtext(side=3,line=1,outer=TRUE,subject[1])
  mtext(side=1,line=1,outer=TRUE,subject[3])
  par(mfrow=c(1,2), oma=c(5,3,5,0))
#
# Plot PDF for X
#
  plot(x,pd,type="h",xlab="X",ylab="",
    sub=subject[3], ylim=c(0,max(pd,td)), err=-1)
  lines(tx,tt)
  mtext(side=2, line=2, outer=T, "PDF")
#
# Plot PDF for Y
#
  yy_rep(y,rep(2,nn))
  hh_rep(hi,rep(2,(nn-1)))
  hh_c(0,hh,0)
  plot(y,c(hi,hi[nn-1]),type="h",
     xlab="Y",ylab="",ylim=c(0,max(hi)),
     sub=encode("Increment of Y :",dd),err=-1)
  lines(yy,hh)
  mtext(side=3,line=1,outer=TRUE,
        encode(subject[1],"",subject[2]))
  mtext(side=1,line=1,outer=TRUE,
  "This is the overall picture for transformation. If you want
  to see the picture one by one, keep going on hitting Return
  when GO? appears until you hit Break Key")
```

```
#
# Iterated Plot
#
  for( i in 1:(nn-1)) {
  par(mfrow=c(1,2), oma=c(5,3,5,0))
#
# PDF plot for X
#
  plot(x[1:( i+1)],pd[1:( i+1)],type="h",
    xlab="X",ylab="",err=-1,
    sub=encode("Increment of X =",x[ i+1]-x[ i]),
    ylim=c(0,max( td[1:(4* i+1)])))
  lines(px[1:(4* i+1)],td[1:(4* i+1)])
  hatch(c(x[ i],px[(4* i-3):(4* i+1)],x[ i+1]),
        c(0,td[(4* i-3):(4* i+1)],0))
  mtext(side=2, line=2, outer=T, "PDF")
#
# PDF plot for Y
#
  yyy_rep(y[1:( i+1)],rep(2,( i+1)))
  hhh_rep(hi[1:i],rep(2,i))
  hhh_c(0,hhh,0)
  plot(y[1:( i+1)],c(hi[1:i],hi[i]),
      type="h",xlab="Y",ylab="",
      ylim=c(0,max(hi[1:i])),err=-1,
      sub=encode("Increment of Y =",dd))
  lines(yyy,hhh)
  hatch(c(y[ i],y[ i],y[ i+1],y[ i+1]),
        c(0,hi[ i],hi[ i],0))
  mtext(side=3,line=1,outer=TRUE,
        encode(subject[1],subject[2],"",subject[3]))
  xv_x[ i] ; dv_pd[ i]
  yv_y[ i] ; hv_hi[ i]
  mtext(side=1,line=1,outer=TRUE,
  encode("X =",xv,"PDF X =",dv,
  "    Y =",yv,"PDF Y =",hv))
     }
  rm(x,y,px,py,dd,td,pd,hi,nn,xv,yv,dv,hv,subject,
     hh,yy,hhh,yyy,tx,ty,tt)
})
END
```

```
MACRO trtplot(x,y,pd,hi,px,py,dd,td,nn,xx,xd,
              rx,rd,subject)
({
#
# X  : X range [-sqrt(e) -sqrt(b)] & [sqrt(b) sqrt(e)]
# Y  : Y range [b e]
# PD : PDF for X
# HI : PDF(estimated) for Y
# PX : X [-s(e) s(e)] with small interval
# PY : Y for PX
# DD : Given interval for Y
# TD : PDF for PX
# NN : length of Y
# XX : X range same as X but with small interval
# XD : PDF for XX
# RX : X range [-s(b) s(b)]
# RD : PDF for RX
# Subject : Related title
#
  par(mfrow=c(1,2),oma=c(5,0,5,0))
#
# Plot X vs. Y
#
  plot(px,py,type="l",xlab="X",ylab="Y")
#
# Plot Y vs. X
#
  plot(py,px,type="l",xlab="Y",ylab="X",err=-1)
  mtext(side=3,line=0,outer=TRUE,subject[1])
  mtext(side=1,line=0,outer=TRUE,subject[3])
  par(mfrow=c(1,2), oma=c(5,3,5,0))
#
# Plot PDF for X
#
  plot(x,pd,type="h",xlab="X",ylab="",
    sub=subject[3], ylim=c(0,max(pd,td)),err=-1)
  lines(px,td)
  mtext(side=2, line=2, outer=T, "PDF")
#
# Plot PDF for Y
#
  yy_rep(y,rep(2,nn))
  hh_rep(hi,rep(2,(nn-1)))
  hh_c(0,hh,0)
  plot(y,c(hi,hi[nn-1]),type="h",
     xlab="Y",ylab="",ylim=c(0,max(hi)),
     sub=encode("Increment of Y :",dd),err=-1)
  lines(yy,hh)
  mtext(side=3,line=1,outer=TRUE,
  encode(subject[1],"  ",subject[2]))
  mtext(side=1,line=1,outer=TRUE,
  "This is the overall picture for transformation. If you want
```

```
  to see the picture one by one, keep going on hitting Return
  when GO? appears until you hit Break Key")
#
# Iterated Plot
#
  for( i in 1:(nn-1)) {
  par(mfrow=c(1,2), oma=c(5,3,5,0))
#
# PDF plot for X
#
  plot(c(x[nn:(nn-i)],x[(nn+1):(nn+1+i)]),
       c(pd[nn:(nn-i)],pd[(nn+1):(nn+1+i)]),type="h",
    xlab="X",ylab="",err=-1,
    sub=encode("Increment of X =",x[nn+1+i]-x[nn+i]),
    ylim=c(0,max(pd[nn:(nn-i)],pd[(nn+1):(nn+1+i)],rd)))
  mx_len(xx)/2
  lines(c(xx[(mx-4*i):mx]),
        c(xd[(mx-4*i):mx]))
  lines(rx,rd)
  lines(xx[(mx+1):(mx+4*i+1)],
        xd[(mx+1):(mx+4*i+1)])
  hatch(c(x[nn+1-i],x[nn-i],xx[(mx-4*i):(mx-4*(i-1))]),
        c(0,0,xd[(mx-4*i):(mx-4*(i-1))]))
  hatch(c(x[nn+1+i],x[nn+i],xx[(mx+4*i-3):(mx+4*i+1)]),
        c(0,0,xd[(mx+4*i-3):(mx+4*i+1)]), angle=135)
  mtext(side=2, line=2, outer=T, "PDF")
#
# PDF plot for Y
#
  yyy_rep(y[1:(i+1)],rep(2,(i+1)))
  hhh_rep(hi[1:i],rep(2,i))
  hhh_c(0,hhh,0)
  plot(y[1:(i+1)],c(hi[1],hi[1:i]),
     type="h",xlab="Y",ylab="",
     ylim=c(0,max(hi[1:i])),err=-1,
     sub=encode("Increment of Y =",dd))
  lines(yyy,hhh)
  hatch(c(y[i],y[i],y[i+1],y[i+1]),
        c(0,hi[i],hi[i],0))
  mtext(side=3,line=1,outer=TRUE,
  encode(subject[1],subject[2],"",subject[3]))
  x1_x[nn+i]   ; dv1_pd[nn+i]
  x2_x[nn+1-i] ; dv2_pd[nn+1-i]
  yv_y[i]      ; hv_hi[i]
  mtext(side=1,line=1,outer=TRUE,
  encode("X :",x2,"PDF =",dv2,"/ X :",x1,"PDF =",dv1))
  mtext(side=1,line=3,outer=TRUE,
  encode("Y =",yv,"PDF Y =",hv))
     }
  rm(x,y,px,py,dd,td,pd,hi,nn,x1,x2,yv,dv1,dv2,hv,
     mx,xx,xd,rx,rd,subject,hh,yy,hhh,yyy)
})
END
```

```
MACRO tr1u(subject,para,lowy,hiy,
ddd/?PROMPT(Number of intervals in Y range  : )/)
#
({
  ?T(i)_ddd
   if(para[1]>=0 ) ?tr11u(subject,para,lowy,hiy,?T(i))
   if(para[2]<=0 ) ?tr12u(subject,para,lowy,hiy,?T(i))
   if(para[1]<0 & para[2]>0) ?tr13u(subject,para,lowy,
                              hiy,?T(i))
})
END
```

```
MACRO tr11u(subject,para,lowy,hiy,int)
#
# Macro for Plotting trnsform of uniform when all parameters
# are greater than zero.
#
({
#
# Y range with given number of intervals
#
  y_seq(lowy,hiy,len=(int+1))
  x_sqrt(y)              # X  : Transformation
  nn_len(y)
#
  pd_dunif(x,para[1],para[2])     # PD : PDF for X
  cd_punif(x,para[1],para[2])     # CD : CDF for X
#
# Area between x[i] and x[i+1], which can be computed
# by subtracting CDF[i] from CDF[i+1].
#
  ay_cd[2:(int+1)]-cd[1:int]
#
# Increment of Y can be computed by Y[i+1]-Y[i]
#
  inc_y[2]-y[1]
#
# Hight for given Y (estimated pdf) is obtained from
# area/intrement.
#
  hi_ay/inc
  subject_c(subject,"Y = X ^2")
#
# Plot X vs Y
#
  par(mfrow=c(1,2), oma=c(5,0,5,0) )
  plot(x,y, type="l",
       xlab="X", ylab="Y")
#
# Plot Y vs. X
#
  plot(y,x,type="l",xlab="Y",ylab="X")
  mtext(side=3,line=0,outer=TRUE,subject[1])
  mtext(side=1,line=0,outer=TRUE,subject[3])
  par(mfrow=c(1,2), oma=c(6,5,0,0))
#
# Plot PDF for X
#
  par(oma=c(5,3,5,0))
  plot(x,pd,type="h",xlab="X",ylab="",
    sub=subject[3], ylim=c(0,max(pd)))
  lines(c(x[1],x[int+1]),c(pd[1],pd[int+1]))
  mtext(side=2, line=2, outer=T, "PDF")
```

```
#
# Plot PDF for Y
#
  yy_rep(y,rep(2,nn))
  hh_rep(hi,rep(2,(nn-1)))
  hh_c(0,hh,0)
  plot(y,c(hi,hi[nn-1]),type="h",
     xlab="Y",ylab="",ylim=c(0,max(hi)),
     sub=encode("Increment of Y :",inc),err=-1)
  lines(yy,hh)
  mtext(side=3,line=1,outer=TRUE,
        encode(subject[1],"",subject[2]))
  mtext(side=1,line=1,outer=TRUE,
  "This is the overall picture for transformation. If you want
  to see the picture one by one, keep going on hitting Return
  when GO? appears until you hit Break Key")
#
# Iterated Plot
#
  for( i in 1:(nn-1)) {
  par(mfrow=c(1,2), oma=c(5,3,5,0))
#
# PDF plot for X
#
  plot(x[1:( i+1)],pd[1:( i+1)],type="h",
    xlab="X",ylab="",err=-1,
    sub=encode("Increment of X =",x[ i+1]-x[ i]),
    ylim=c(0,pd))
  lines(x[1:( i+1)],pd[1:( i+1)])
  hatch(c(x[ i],x[ i],x[ i+1],x[ i+1]),
        c(0,pd[ i],pd[ i],0))
  mtext(side=2, line=2, outer=T, "PDF")
#
# PDF plot for Y
#
  yyy_rep(y[1:( i+1)],rep(2,( i+1)))
  hhh_rep(hi[1: i],rep(2, i))
  hhh_c(0,hhh,0)
  plot(y[1:( i+1)],c(hi[1: i],hi[ i]),
     type="h",xlab="Y",ylab="",
     ylim=c(0,max(hi[1: i])),err=-1,
     sub=encode("Increment of Y =", inc))
  lines(yyy,hhh)
  hatch(c(y[ i],y[ i],y[ i+1],y[ i+1]),
        c(0,hi[ i],hi[ i],0))
  mtext(side=3,line=1,outer=TRUE,
        encode(subject[1],subject[2],"",subject[3]))
  xv_x[ i] ; dv_pd[ i]
  yv_y[ i] ; hv_hi[ i]
  mtext(side=1,line=1,outer=TRUE,
```

```
  encode("X =",xv,"PDF X =",dv,
  "   Y =",yv,"PDF Y =",hv))
      }
  rm(x,y,pd,cd,hi,nn,xv,yv,dv,hv,subject,
     hh,yy,hhh,yyy, inc)
})
END
```

```
MACRO tr12u(subject,para,lowy,hiy,int)
#
# Macro for Plotting trnsform of uniform when all parameters
# are smaller than zero.
#
({
#
# Y range with given number of intervals
#
  y_seq(lowy,hiy,len=(int+1))
  x_sqrt(y)            # X  : Transformation
  x_-rev(x)
  nn_len(y)
#
  pd_dunif(x,para[1],para[2])     # PD : PDF for X
  cd_punif(x,para[1],para[2])     # CD : CDF for X
#
# Area between x[i] and x[i+1], which can be computed
# by subtracting CDF[i] from CDF[i+1].
#
  ay_cd[2:(int+1)]-cd[1:int]
#
# Increment of Y can be computed by Y[i+1]-Y[i]
#
  inc_y[2]-y[1]
#
# Hight for given Y (estimated pdf) is obtained from
# area/intrement.
#
  hi_ay/inc
  hi_rev(hi)
  subject_c(subject,"Y = X ^2")
#
# Plot X vs Y
#
  par(mfrow=c(1,2), oma=c(5,0,5,0) )
  plot(x,y[nn:1], type="l", err=-1,
       xlab="X", ylab="Y")
#
# Plot Y vs. X
#
  plot(y,x[nn:1],type="l",xlab="Y",ylab="X")
  mtext(side=3,line=0,outer=TRUE,subject[1])
  mtext(side=1,line=0,outer=TRUE,subject[3])
  par(mfrow=c(1,2), oma=c(5,3,5,0))
```

```
#
# Plot PDF for X
#
  plot(x,pd,type="h",xlab="X",ylab="",
    sub=subject[3], ylim=c(0,max(pd)))
  lines(c(x[1],x[int+1]),c(pd[1],pd[int+1]))
  mtext(side=2, line=2, outer=T, "PDF")
#
# Plot PDF for Y
#
  yy_rep(y,rep(2,nn))
  hh_rep(hi,rep(2,(nn-1)))
  hh_c(0,hh,0)
  plot(y,c(hi,hi[nn-1]),type="h",
     xlab="Y",ylab="",ylim=c(0,max(hi)),
     sub=encode("Increment of Y :",inc),err=-1)
  lines(yy,hh)
  mtext(side=3,line=1,outer=TRUE,
        encode(subject[1],"",subject[2]))
  mtext(side=1,line=1,outer=TRUE,
  "This is the overall picture for transformation. If you want
  to see the picture one by one, keep going on hitting Return
  when GO? appears until you hit Break Key")
#
# Iterated Plot
#
  for( i in 1:(nn-1)) {
  par(mfrow=c(1,2), oma=c(5,3,5,0))
#
# PDF plot for X
#
  plot(x[nn:(nn-i)],pd[nn:(nn-i)],type="h",
    xlab="X",ylab="",err=-1,
    sub=encode("Increment of X =",x[nn-i+1]-x[nn-i]),
    ylim=c(0,pd))
  lines(x[nn:(nn-i)],pd[nn:(nn-i)])
  hatch(c(x[nn-i+1],x[nn-i+1],x[nn-i],x[nn-i]),
        c(0,pd[nn-i+1],pd[nn-i+1],0))
  mtext(side=2, line=2, outer=T, "PDF")
#
# PDF plot for Y
#
  yyy_rep(y[1:(i+1)],rep(2,(i+1)))
  hhh_rep(hi[1:i],rep(2,i))
  hhh_c(0,hhh,0)
  plot(y[1:(i+1)],c(hi[1:i],hi[i]),
     type="h",xlab="Y",ylab="",
     ylim=c(0,max(hi[1:i])),err=-1,
     sub=encode("Increment of Y =",inc))
```

```
  lines(yyy,hhh)
  hatch(c(y[i],y[i],y[i+1],y[i+1]),
        c(0,hi[i],hi[i],0))
  mtext(side=3,line=1,outer=TRUE,
        encode(subject[1],subject[2],"",subject[3]))
  xv_x[nn-i+1] ; dv_pd[nn-i+1]
  yv_y[i] ; hv_hi[i]
  mtext(side=1,line=1,outer=TRUE,
  encode("X =",xv,"PDF X =",dv,
  "   Y =",yv,"PDF Y =",hv))
     }
  rm(x,y,pd,cd,hi,nn,xv,yv,dv,hv,subject,
     hh,yy,hhh,yyy,inc)
})
END
```

```
MACRO tr13u(subject,para,lowy,hiy,int)
#
# Macro for Plotting trnsform of uniform when lower bound is
# less than zero and upper bound is greater than zero.
#
({
#
# Y range with given number of intervals
#
  y_seq(lowy,hiy,len=(int+1))
  x_sqrt(y)              # X  : Transformation
  x_c(-rev(x),x)
  x_x[x>para[1] & x<para[2]]
  x_c(para[1], x, para[2])
  x_uniq(x)
  ox_?which(x==0)
  nx_len(x)
  ny_len(y)
  px_seq(min(x),max(x),len=21)
  py_px^2
#
  pd_dunif(x,para[1],para[2])     # PD : PDF for X
  cd_punif(x,para[1],para[2])     # CD : CDF for X
#
# Area between x[i] and x[i+1], which can be computed
# by subtracting CDF[i] from CDF[i+1].
#
  ayx_cd[2:nx]-cd[1:(nx-1)]
#
# Increment of Y can be computed by Y[i+1]-Y[i]
#
  inc_y[2]-y[1]
#
# Some range of X should be added up for Y range
#
  if( abs(para[1]) > para[2] ) {
    ay_ayx[(ox-1):1]
    ay[1:(nx-ny)]_ay[1:(nx-ny)]+ayx[ox:(nx-1)]
    ix_x[ny:2]-x[(ny-1):1]
    }
  if( abs(para[1]) <= para[2] ) {
    ay_ayx[ox:(nx-1)]
    ay[1:(ox-1)]_ay[1:(ox-1)]+ayx[(ox-1):1]
    ix_x[(nx-ny+2):nx]-x[(nx-ny+1):(nx-1)]
    }
#
# Hight for given Y (estimated pdf) is obtained from
# area/intrement.
#
  hi_ay/inc
  subject_c(subject,"Y = X ^2")
```

```
#
# Plot X vs Y
#
  par(mfrow=c(1,2), oma=c(5,0,5,0) )
  plot(px,py, type="l",
       xlab="X", ylab="Y")
#
# Plot Y vs. X
#
  plot(py,px,type="l",xlab="Y",ylab="X")
  mtext(side=3,line=0,outer=TRUE,subject[1])
  mtext(side=1,line=0,outer=TRUE,subject[3])
  par(mfrow=c(1,2), oma=c(5,3,5,0))
#
# Plot PDF for X
#
  plot(x,pd,type="h",xlab="X",ylab="",
    sub=subject[3], ylim=c(0,max(pd)))
  lines(c(x[1],x[nx]),c(pd[1],pd[nx]))
  mtext(side=2, line=2, outer=T, "PDF")
#
# Plot PDF for Y
#
  yy_rep(y,rep(2,ny))
  hh_rep(hi,rep(2,(ny-1)))
  hh_c(0,hh,0)
  plot(y,c(hi,hi[ny-1]),type="h",
     xlab="Y",ylab="",ylim=c(0,max(hi)),
     sub=encode("Increment of Y :",inc),err=-1)
  lines(yy,hh)
  mtext(side=3,line=1,outer=TRUE,
        encode(subject[1],"",subject[2]))
  mtext(side=1,line=1,outer=TRUE,
  "This is the overall picture for transformation. If you want
  to see the picture one by one, keep going on hitting Return
  when GO? appears until you hit Break Key")
#
# Iterated Plot
#
  for( i in 1:(ny-1)) {
  par(mfrow=c(1,2), oma=c(5,3,5,0))
#
# PDF plot for X
#
  plot(x[max(1,(ox-i)):min(nx,ox+i)],
       pd[max(1,(ox-i)):min(nx,ox+i)], type="h",
    sub=encode("Increment of X =",ix[i]),
    ylim=c(0,pd), xlab="X", ylab="")
  lines(c(x[max(1,(ox-i))],x[min(nx,ox+i)]), rep(pd[ox],2))
```

```
  if( ox-i >=1 ) {
  hatch(c(x[max(1,(ox-i)+1)], x[max(1,(ox-i)+1)],
          x[max(1,(ox-i))], x[max(1,(ox-i))]),
        c(0,pd[ox],pd[ox],0))        }
  if( ox+i <= nx ) {
  hatch(c(x[min(nx,(ox+i-1))], x[min(nx,(ox+i-1))],
          x[min(nx,ox+i)], x[min(nx,ox+i)]),
        c(0,pd[ox],pd[ox],0), angle=135)       }
  mtext(side=2, line=2, outer=T, "PDF")
#
# PDF plot for Y
#
  yyy_rep(y[1:(i+1)],rep(2,(i+1)))
  hhh_rep(hi[1:i],rep(2,i))
  hhh_c(0,hhh,0)
  plot(y[1:(i+1)],c(hi[1:i],hi[i]),
     type="h",xlab="Y",ylab="",
     ylim=c(0,max(hi[1:i])),err=-1,
     sub=encode("Increment of Y =",inc))
  lines(yyy,hhh)
  hatch(c(y[i],y[i],y[i+1],y[i+1]),
        c(0,hi[i],hi[i],0))
  mtext(side=3,line=1,outer=TRUE,
        encode(subject[1],subject[2],"",subject[3]))
  x1_ifelse(ox+i<=nx, x[min((ox+i-1),nx)], NA)
  dv1_ifelse(ox+i<=nx, pd[ox], 0)
  x2_ifelse((ox-i)>=1, x[max((ox-i+1),1)], NA)
  dv2_ifelse((ox-i)>=1, pd[ox], 0)
  mtext(side=1,line=1,outer=TRUE,
  encode("X :",x2,"PDF =",dv2,"/ X :",x1,"PDF =",dv1))
  mtext(side=1,line=3,outer=TRUE,
  encode("Y =",y[i],"PDF Y =",hi[i]))     }
  rm(x,y,pd,cd,hi,ny,nx,subject,ox,ix,ay,ayx,
     hh,yy,hhh,yyy,inc,px,py,para,lowy,hiy,int)
})
END
```

```
MACRO unifins1(lowy,hiy)
({
   message(encode("Lower Y limit is",lowy,
          " Upper Y limit is", hiy))
})
END
```

```
MACRO unifins2(lowy,hiy)
({
   message(encode(" Possible Lower Y limit is",lowy,
          " Upper Y limit is", hiy))
})
END
```

```
MACRO bvtran
({
  ?MESSAGE()
  ?MESSAGE(Type of Distribution :)
  ?MESSAGE()
  item_c("Bivariate Normal Distribution",
         "Bivariate Uniform(0,1) Distribution",
         "Bivariate Chisquare Distribution",
         "Standard Normal & Chisquare Distribution")
  action_c("?bvntran",
           "?bvutran",
           "?bvctran",
           "?bvnctran")
  menu( item, action)
  rm( item, action)
})
END
```

```
MACRO bvntran
({
  ?MESSAGE()
  ?MESSAGE(Type of Transformation :)
  ?MESSAGE()
  item_c("z = x + y",
         "z = x^2 + y^2")
  action_c("?bvn1tran",
           "?bvn2tran")
  menu( item, action)
  rm( item, action)
})
END
```

```
MACRO bvutran
({
  ?MESSAGE()
  ?MESSAGE(Type of Transformation :)
  ?MESSAGE()
  item_c("z = x + y",
         "z = x^2 + y^2")
  action_c("?bvu1tran",
           "?bvu2tran")
  menu( item, action)
  rm( item, action)
})
END
```

```
MACRO bvctran(
d1/?PROMPT(Degree of Freedom for X   : )/,
d2/?PROMPT(Degree of Freedom for Y   : )/,
it/?PROMPT(Number of intervals you want to divide the Z range : )/)
chapter
({
#  This is a Macro to transform bivariate chisquare
#  variables with the form of z = (x/dfx) / (y/dfy)
#
   ?MESSAGE()
   ?MESSAGE(Bivariate Chisquare variable will be transformed)
   ?MESSAGE(as Z = (X/DFx) / (Y/DFy))
   ?MESSAGE()
   ?T(d1)_d1
   ?T(d2)_d2
   ?T(dr)_?T(d1)/?T(d2)
#
#  Range of Z will be from 0 to 7,
#  because it will give clear plotting.
#
   ?MESSAGE()
   ?MESSAGE(Note :)
   ?MESSAGE(The range of Z will be from 0 to 7.)
   ?MESSAGE()
   ?T(i)_it
   ?T(z)_seq(0,7,len=(?T(i)+1))
#  To set the range of X
   ?T(x)_seq(0.0001,2,len=10)
#  To set the range of Y
   ?T(y)_?T(x)
#
#  Call macro to plot the z=x+y (3 dimensional plot) and
#  the range of z in terms of x and y dimension.
#
   ?bvcplot1(?T(x),?T(y),?T(d1),?T(d2),?T(dr))
   ?T(xr)_rep(?T(x)*?T(d1),rep(10,10))
   ?T(yr)_rep(?T(y)*?T(d2),10)
#
#  Compute the pdf to plot the contour and the 3-dimensional
#  picture.
#
   ?T(w)_dchisq(?T(xr),?T(d1)) * dchisq(?T(yr),?T(d2))
   ?T(w)_matrix(?T(w),10,10,byrow=T)
#
#  Call macro to plot contour and 3-dimensional picture.
#
   ?bvcplot2(?T(x),?T(y),?T(w),?T(d1),?T(d2),?T(dr))
```

```
#
#  Compute the volume under the given area by calling the imsl
#  library for double integration via S extention function.
#
   ?T(gx)_gamma(?T(d1)/2)
   ?T(gy)_gamma(?T(d2)/2)
#
#  yh : max value for y given degree of freedom
#  This value will be used for the max range of outer interval
#  of double integration.
#
   ?T(yh)_qchisq(.9999,?T(d2))
   ?T(cd)_bvchisq(?T(d1),?T(d2),?T(i),?T(gx),?T(gy),?T(yh),?T(z))
#
#  Call macro to plot the estimated pdf for transformed variable z,
#  and the pdf contour for x and y.
#
   ?bvcplot3(?T(cd),?T(z),?T(i),?T(x),?T(y),?T(w),
          ?T(d1),?T(d2),?T(dr))
   rm(?T(d1),?T(d2),?T(z),?T(i),?T(x),?T(y),?T(w),?T(xr),?T(yr),
      ?T(gx),?T(gy),?T(yh),?T(cd),?T(dr))
})
END
```

```
MACRO bvnctran(
df/?PROMPT(Degree of Freedom for Chisquare variable : )/,
it/?PROMPT(Number of intervals you want to divide the T range : )/)
chapter
({
#  This is a Macro to transform Standard Normal variable and
#  Chisquare variable into T variable.
#  The transformation equation is T = X / (sqrt(Y/n)),
#     where,  X : Normal variable with mu=0 sd=1
#             Y : Chisquare variable with n degree of freedom
#
   ?MESSAGE()
   ?MESSAGE(X  : Normal variable with Mu = 0  SD = 1)
   ?MESSAGE(Yp : Chisquare variable with n degree of freedom)
   ?MESSAGE(The transform will be  T = X / sqrt[Yp/n] )
   ?MESSAGE()
   ?T(df)_df
#
#  Range of T will be from -4 to 4,
#  because it will give clear plotting.
#
   ?MESSAGE()
   ?MESSAGE(Note :)
   ?MESSAGE(The range of T will be from -4 to 4.)
   ?MESSAGE()
   ?T(i)_it
   ?T(z)_seq(-4,4,len=(?T(i)+1))
#  To set the range of X
   ?T(x)_seq(-3,3,len=11)
#  Quantile of chisquare varible with .9999 cdf
   ?T(mc)_qchisq(.9999, ?T(df))
#  To set the range of Yp
   ?T(yp)_seq(0.0000001,?T(mc),len=11)
#  To find the value of Y=sqrt(Yp/n)
   ?T(y)_sqrt(?T(yp)/?T(df))
#  Max value of yp
   ?T(my)_max(?T(y))
#
#  Call macro to plot the T=X/Y or T=X/sqrt(Yp/n)
#  (3 dimensional plot) and the range of T in terms of
#  x and y dimension.
#
   ?bvncplot1(?T(x),?T(y),?T(df),?T(my))
#
#  Repeat X and YP values to compute PDF
#
   ?T(xr)_rep(?T(x), rep(11,11))
   ?T(yr)_rep(?T(yp), 11)
```

```
#
#  Compute the pdf to plot the contour and the 3-dimensional
#  picture.
#
   ?T(w)_dnorm(?T(xr)) * dchisq(?T(yr),?T(df))
   ?T(w)_matrix(?T(w),11,11,byrow=T)
#
#  Call macro to plot contour and 3-dimensional picture.
#
   ?bvncplot2(?T(x),?T(y),?T(w),?T(df),?T(my))
#
   ?T(gx)_gamma(?T(df)/2)
#
#  Compute the volume under the given area by calling the imsl
#  library for double integration via S extention function.
#
   ?T(cd)_bvnandc(?T(df),?T(i),?T(gx),?T(mc),?T(z))
#
#  Call macro to plot the estimated pdf for transformed variable t,
#  and the pdf contour for x and y.
#
   ?bvncplot3(?T(cd),?T(z),?T(i),?T(x),?T(y),?T(w),?T(df),?T(my))
#  rm(?T(df),?T(z),?T(i),?T(x),?T(y),?T(yp),?T(w),?T(xr),?T(yr),
#     ?T(gx),?T(yh),?T(cd),?T(mc),?T(my))
})
END
```

```
MACRO bvn1tran(
mu/?PROMPT(Mean value   : )/,
sg/?PROMPT(Sigma value  : )/,
rv/?PROMPT(Coef of R    : )/,
it/?PROMPT(Number of intervals you want to divide the Z range : )/)
chapter
({
#  This is a Macro to transform bivariate normal
#  variables with the form of z = x + y
#
   ?MESSAGE()
   ?MESSAGE(Bivariate normal variables will be drawn from)
   ?MESSAGE( identical distribution. )
   ?MESSAGE()
   ?T(m)_mu
   ?T(s)_sg
   ?T(r)_rv
   ?T(z1)_(?T(m)-2*?T(s))*2
   ?T(z2)_(?T(m)+2*?T(s))*2
#
#  Range of Z will be from (Mu-2*S)*2 to (Mu+2*S)*2,
#  because it will give clear plotting.
#
   ?MESSAGE()
   ?MESSAGE(Note :)
   ?MESSAGE(The range of Z will be from (Mu-2*S)*2 to (Mu+2*S)*2.)
   ?MESSAGE()
   ?T(i)_it
   ?T(z)_seq(?T(z1),?T(z2),len=(?T(i)+1))
   ?T(x1)_?T(m)-3*?T(s)
   ?T(x2)_?T(m)+3*?T(s)
#
#  nx & mx will represent integer value for the range of
#  x and y.
#
   ?T(nx)_(floor(?T(x1)/5))*5
   ?T(mx)_(ceiling(?T(x2)/5))*5
#
#  Call macro to plot the z=x+y (3 dimensional plot) and
#  the range of z in terms of x and y dimension.
#
   ?bvn1plot1(?T(z1),?T(z2),?T(m),?T(s),?T(x1),?T(x2),?T(nx),?T(mx))
#  To set the range of X
   ?T(x)_seq(?T(x1),?T(x2),len=11)
#  To set the range of Y
   ?T(y)_?T(x)
   ?T(xr)_rep(?T(x),rep(11,11))
   ?T(yr)_rep(?T(y),11)
```

```
#
#  Compute the pdf to plot the contour and the 3-dimensional
#  picture.
#
   ?T(w)_?dbinorm1(?T(m),?T(s),?T(m),?T(s),?T(r),?T(xr),?T(yr))
   ?T(w)_matrix(?T(w),11,11,byrow=T)
#
#  Call macro to plot contour and 3-dimensional picture.
#
   ?bvn1plot2(?T(x),?T(y),?T(w),?T(m),?T(s),?T(r),?T(z1),?T(z2),
             ?T(nx),?T(mx),?T(x1),?T(x2))
#
#  Compute the volume under the given area by calling the imsl
#  library for double integration via S extention function.
#
  ?T(cd)_bvnorm(?T(m),?T(s),?T(m),?T(s),?T(r),?T(i),?T(z))
#
#  Call macro to plot the estimated pdf for transformed variable z,
#  and the pdf contour for x and y.
#
   ?bvn1plot3(?T(cd),?T(z),?T(z1),?T(z2),?T(i),?T(x),?T(y),?T(w),
             ?T(nx),?T(mx),?T(m),?T(s),?T(r))
   rm(?T(m),?T(s),?T(z1),?T(z2),?T(i),?T(r),?T(cd),
      ?T(x),?T(y),?T(w),?T(xr),?T(yr),?T(z))
})
END
```

```
MACRO bvn1plot1(z1,z2,m,s,x1,x2,nx,mx)
#
# This macro is to plot relationship between z and x,y and
# the range of z in terms of x and y.
#
({
  x_seq(x1,x2,len=11)
  y_x
  xx_rep(x,rep(11,11))
  yy_rep(y,11)
  xx_matrix(xx,11,11,byrow=T)
  yy_matrix(yy,11,11,byrow=T)
  zz_xx+yy
  nz_min(zz)
  mz_max(zz)
  zz_zz-nz
  par(mfrow=c(1,2), oma=c(4,0,4,0))
#
# Plot the "z" in the 3 dimensinal plane.
#
  persp(zz/mz)
  title(sub=encode("Y(L) & X(R)  [",x1,x2,"]"))
#
# Plot the x and y axis
#
  plot(x,y, xlim=c(nx,mx), ylim=c(nx,mx), type="n", axes=F)
  axis(1) ; axis(2)
  title(sub=encode("Given Mu :",m,"  S :",s))
#
# Plot the line representing Mean value.
#
  abline(h=m)
  abline(v=m)
#
# Plot the line representing starting and ending z.
#
  abline(z1,-1)
  abline(z2,-1)
#
# Shade the z range.
#
  hatch(c(nx,nx,(z2-mx),mx,mx,(z1-nx)),
        c((z1-nx),mx,mx,(z2-mx),nx,nx),
        border=F)
  mtext(side=3, line=1, outer=T,
    "Plot of Z = X + Y  and  Range of Transform")
  mtext(side=1, line=1, outer=T,
    encode("Starting Z:",z1,"  Ending Z:",z2))
  rm(x,y,xx,yy,zz,nz,mz)
})
END
```

```
MACRO bvn1plot2(x,y,z,m,s,r,z1,z2,nx,mx,x1,x2)
#
# This macro is to plot the pdf contour and the pdf plane
# in 3 dimensional plane.
#
({
  par(mfrow=c(1,2), oma=c(4,0,4,0))
  mz_max(z)
#
# Plot the contour
#
  contour(x,y,z, xlim=c(nx,mx), ylim=c(nx,mx), axes=F,
          xlab="X", ylab="Y")
  axis(1) ;  axis(2)
#
# Plot the starting and ending z on the contour plot.
#
  abline(z1,-1)
  abline(z2,-1)
#
# Shade the z range.
#
  hatch(c(nx,nx,(z2-mx),mx,mx,(z1-nx)),
        c((z1-nx),mx,mx,(z2-mx),nx,nx),
        border=F)
  title(sub=encode("Starting Z:",z1,"  Ending Z:",z2))
#
# Plot the pdf plane in 3 dimensional plane.
#
  persp(z/mz)
  title(sub=encode("Y(L) & X(R) : [",x1,x2,"]"))
  mtext(side=3, line=1, outer=T,
    "Bivariate  Normal PDF Surface")
  mtext(side=1, line=1, outer=T,
    encode("Given Mu :",m,"  S :",s," R :",r))
  rm(mz)
})
END
```

```
MACRO bvn1plot3(cd,z,z1,z2,i,x,y,w,nx,mx,m,s,r)
#
# This macro is to plot transformed pdf and the original
# pdf contour.
#
({
  par(mfrow=c(1,2), oma=c(5,3,4,0) )
  zinc_z[2]-z[1]
  pd_cd/zinc
  zz_rep(z,rep(2,(i+1)))
  hi_rep(pd,rep(2,i))
  hi_c(0,hi,0)
  totpdf_sum(cd)
#
# Plot transformed pdf.
#
  plot(z[1:i],pd, type="h", xlab="Z", ylab="",
       ylim=c(0,max(hi)), xlim=c(z[1],z[i+1]),
       main=encode("Total PDF =",totpdf),
       sub=encode("Increment of Z :",zinc), err=-1)
  lines(zz,hi)
  mtext(side=2, line=1, outer=T, "PDF")
#
# Pot pdf contour in x-y plane.
#
  contour(x,y,w, xlim=c(nx,mx), ylim=c(nx,mx), axes=F,
          xlab="X", ylab="Y")
  axis(1) ;  axis(2)
  for ( j in 1:(i+1) ) {
    abline(z[j],-1)
    }
  title(sub=encode("Starting Z:",z1,"  Ending Z:",z2))
  mtext(side=3, line=1, outer=T,
    "Transform of Bivariate Normal variable with Z = X + Y")
  mtext(side=1, line=1, outer=T,
    encode("Given Mu :",m,"  S :",s," R :",r))
  mtext(side=1, line=3, outer=T,
    "This is the overall picture for transformation. If you want to see
     the picture one by one, hit Return at every time GO? appears")
#
# Iterative plot
#
  par(oma=c(6,0,3,0), oma=c(5,3,4,0))
  for( j in 1:i ) {
  totpdf_sum(cd[1:j])
```

```
#
# Plot cumulative transformed pdf.
#
  plot(c(z[1],z[2:(j+1)]),c(pd[1],pd[1:j]), type="h", xlab="Z",
        ylab="", ylim=c(0,max(pd[1:j])), xlim=c(z[1],z[j+1]),
        main=encode("Total PDF =",totpdf),
        sub=encode("Increment of Z :", zinc), err=-1)
  lines(zz[1:(1+2*j)], hi[1:(1+2*j)])
  hatch(c(z[j],z[j],z[j+1],z[j+1]),
        c(0,pd[j],pd[j],0))
  mtext(side=2, line=1, outer=T, "PDF")
#
# Plot cumulative pdf contour in x-y plane.
#
  contour(x,y,w, xlim=c(nx,mx), ylim=c(nx,mx), axes=F,
          xlab="X", ylab="Y")
  axis(1) ;  axis(2)
  for ( k in 0:j ) {
     abline(z[k+1],-1)
     }
#
# Compute the coordinates for the current plot and shade the
# current area in contour plot.
#
  x1_max(nx,z[j]-mx)
  x2_max(nx,z[j+1]-mx)
  x3_min(mx,z[j+1]-nx)
  x4_min(mx,z[j]-nx)
  hatch(c(x1,x2,x3,x4),c(x4,x3,x2,x1),border=F)
  title(sub=encode("Starting Z:",z[j],"  Ending Z:",z[j+1]))
  mtext(side=3, line=1, outer=T,
   "Transform of Bivariate Normal variable with Z = X + Y")
  mtext(side=1, line=1, outer=T,
  encode("Given Mu :",m,"  S :",s," R :",r))
  mtext(side=1, line=3, outer=T,
  encode("When Z =",z[j],"   PDF =",pd[j]))
   }
  rm(zinc,pd,zz,hi,x1,x2,x3,x4,totpdf)
})
END
```

```
MACRO bvn2tran(
mu/?PROMPT(Mean value   : )/,
sg/?PROMPT(Sigma value  : )/,
rv/?PROMPT(Coef of R    : )/,
it/?PROMPT(Number of intervals you want to divide the Z range : )/)
chapter
({
#  This is a Macro to transform bivariate normal
#  variables with the form of z = x^2 + y^2
#
   ?MESSAGE(Bivariate normal variables will be drawn from)
   ?MESSAGE( identical distribution.    For best result,)
   ?MESSAGE(-5 <= Mu <= 5,   Sigma <=3)
   ?MESSAGE()
   ?T(m)_mu
   ?T(s)_sg
   ?T(r)_rv
   ?T(z1)_( max(0,(?T(m)-3*?T(s))) )^2
   ?T(z2)_(?T(m)+3*?T(s))^2
#
#  Range of Z will be from (max(0,(Mu-3*S)))^2 to (Mu+3*S)^2,
#  because it will give clear plotting.
#
   ?MESSAGE()
   ?MESSAGE(Note :)
   ?MESSAGE(The range of Z will be from
           (max(0,(Mu-3*S)))^2 to (Mu+3*S)^2.)
   ?MESSAGE()
   ?T(i)_it
   ?T(z)_seq(?T(z1),?T(z2),len=(?T(i)+1))
   ?T(zs)_sqrt(?T(z))
   ?T(x1)_?T(m)-3*?T(s)
   ?T(x2)_?T(m)+3*?T(s)
   ?T(mx)_max(abs(?T(x1)),abs(?T(x2)))
   ?T(cx)_rep(0,?T(i)+1)
#
#  Call macro to plot the z=x^2+y^2 (3 dimensional plot) and
#  the range of z in terms of x and y dimension.
#
   ?bvn2plot1(?T(z1),?T(z2),?T(zs),?T(cx),?T(m),?T(s),
              ?T(x1),?T(x2),?T(mx))
#  To set the range of X
   ?T(x)_seq(?T(x1),?T(x2),len=11)
#  To set the range of Y
   ?T(y)_?T(x)
   ?T(xr)_rep(?T(x),rep(11,11))
   ?T(yr)_rep(?T(y),11)
```

```
#
#  Compute the pdf to plot the contour and the 3-dimensional
#  picture.
#
   ?T(w)_?dbinorm1(?T(m),?T(s),?T(m),?T(s),?T(r),?T(xr),?T(yr))
   ?T(w)_matrix(?T(w),11,11,byrow=T)
#
#  Call macro to plot contour and 3-dimensional picture.
#
   ?bvn2plot2(?T(x),?T(y),?T(w),?T(m),?T(s),?T(r),?T(z),?T(zs),
            ?T(mx),?T(cx),?T(i),?T(x1),?T(x2))
#
#  Compute the volume under the given area by calling the imsl
#  library for double integration via S extention function.
#
  ?T(cd)_bvnorm2(?T(m),?T(s),?T(m),?T(s),?T(r),?T(i),?T(z))
#
#  Call macro to plot the estimated pdf for transformed variable z,
#  and the pdf contour for x and y.
#
   ?bvn2plot3(?T(cd),?T(z),?T(zs),?T(i),?T(x),?T(y),?T(w),
            ?T(m),?T(s),?T(r),?T(mx))
   rm(?T(m),?T(s),?T(z1),?T(z2),?T(zs),?T(i),?T(r),?T(cd),
      ?T(x),?T(y),?T(w),?T(xr),?T(yr),?T(z),?T(cx),?T(mx))
})
END
```

```
MACRO bvn2plot1(z1,z2,zs,cx,m,s,x1,x2,mx)
#
# This macro is to plot relationship between z and x,y and
# the range of z in terms of x and y.
#
({
  x_seq(x1,x2,len=11)
  xp_c(-mx,mx)
  y_x
  xx_rep(x,rep(11,11))
  yy_rep(y,11)
  xx_matrix(xx,11,11,byrow=T)
  yy_matrix(yy,11,11,byrow=T)
  zz_xx^2+yy^2
#
# Plot the "z" in the 3 dimensinal plane.
#
  mz_max(zz)
  par(mfrow=c(1,2), oma=c(4,0,4,0))
  persp(zz/mz)
  title(sub=encode("Y(L) & X(R)  [",x1,x2,"]"))
#
# Plot the x and y axis
#
  plot(xp,xp,type="n", xlab="X", ylab="Y")
  title(sub=encode("Given Mu :",m,"  S :",s))
#
# Plot the line representing Mean value.
#
  abline(v=m)
  abline(h=m)
#
# Plot the circle representing range of z.
#
  symbols(cx,cx,circle=zs, add=T, inches=F, err=-1)
  mtext(side=3, line=1, outer=T,
    "Plot of Z = X^2 + Y^2  and  Range of Transform")
  mtext(side=1, line=1, outer=T,
    encode("Starting Z:",z1,"  Ending Z:",z2))
  rm(x,y,xx,yy,zz,mz,xp)
})
END
```

```
MACRO bvn2plot2(x,y,w,m,s,r,z,zs,mx,cx,it,x1,x2)
#
# This macro is to plot the pdf contour and the pdf plane
# in 3 dimensional plane.
#
({
  par(mfrow=c(1,2), oma=c(4,0,4,0))
  mw_max(w)
#
# Plot the pdf contour.
#
  contour(x,y,w, xlim=c(-mx,mx), ylim=c(-mx,mx),
          xlab="X", ylab="Y")
#
# Plot the circle representing range z in pdf contour plot.
#
  symbols(cx,cx,circle=zs, add=T, inches=F, err=-1)
  title(sub=encode("Starting Z:",z[1],"  Ending Z:",z[it+1]))
#
# Plot the pdf plane in 3 dimensional plane.
#
  persp(w/mw)
  title(sub=encode("Y(L) & X(R) : [",x1,x2,"]"))
  mtext(side=3, line=1, outer=T,
    "Bivariate  Normal PDF Surface")
  mtext(side=1, line=1, outer=T,
    encode("Given Mu :",m,"  S :",s," R :",r))
  rm(mw)
})
END
```

```
MACRO bvn2plot3(cd,z,zs,i,x,y,w,m,s,r,mx)
#
# This macro is to plot transformed pdf and the original
# pdf contour.
#
({
  par(mfrow=c(1,2), oma=c(5,3,4,0) )
  zinc_z[2]-z[1]
  cd_cd[2:(i+1)]-cd[1:i]
  pd_cd/zinc
  zz_rep(z,rep(2,(i+1)))
  hi_rep(pd,rep(2,i))
  hi_c(0,hi,0)
  totpdf_sum(cd)
#
# Plot the transformed pdf.
#
  plot(z[1:i],pd, type="h", xlab="Z", ylab="",
       ylim=c(0,max(hi)), xlim=c(z[1],z[i+1]),
       main=encode("Total PDF =",totpdf),
       sub=encode("Increment of Z :",zinc), err=-1)
  lines(zz,hi)
  mtext(side=2, line=1, outer=T, "PDF")
#
# Plot the pdf contour in x-y plane
#
  contour(x,y,w, xlim=c(-mx,mx), ylim=c(-mx,mx),
          xlab="X", ylab="Y")
#
# Plot the circle representing range z on the contour
# plot.
#
  symbols(c(0,0),c(0,0),circle=c(zs[1],zs[i+1]),
          add=T, inches=F, err=-1)
  title(sub=encode("Starting Z:",z[1],"  Ending Z:",z[i+1]))
  mtext(side=3, line=1, outer=T,
    "Transform of Bivariate Normal variable with Z = X^2 + Y^2")
  mtext(side=1, line=1, outer=T,
    encode("Given Mu :",m,"  S :",s," R :",r))
  mtext(side=1, line=3, outer=T,
    "This is the overall picture for transformation. If you want to see
     the picture one by one, hit Return at every time GO? appears")
#
# Iterative plot.
#
  for( j in 1:i ) {
```

```
#
# Plot cumulative transformed pdf.
#
  par(mfrow=c(1,2), oma=c(5,3,4,0))
   totpdf sum(cd[1:j])
   plot(c(z[1],z[2:(j+1)]),c(pd[1],pd[1:j]), type="h", xlab="Z",
        ylab="", ylim=c(0,max(pd[1:j])), xlim=c(z[1],z[j+1]),
        main=encode("Total PDF =",totpdf),
        sub=encode("Increment of Z :", zinc), err=-1)
   lines(zz[1:(1+2*j)], hi[1:(1+2*j)])
   hatch(c(z[j],z[j],z[j+1],z[j+1]),
         c(0,pd[j],pd[j],0))
  mtext(side=2, line=1, outer=T, "PDF")
#
# Plot cumulative pdf plane in x-y plane.
#
   contour(x,y,w, xlim=c(-mx,mx), ylim=c(-mx,mx),
           xlab="X", ylab="Y")
#
# Plot the circle representing the current z range on
# the contour plot.
#
   symbols(c(0,0),c(0,0),circle=c(zs[j],zs[j+1]), add=T, inches=F,
           err=-1)
   title(sub=encode("Starting Z:",z[j]," Ending Z:",z[j+1]))
   mtext(side=3, line=1, outer=T,
    "Transform of Bivariate Normal variable with Z = X^2 + Y^2")
   mtext(side=1, line=1, outer=T,
    encode("Given Mu :",m," S :",s," R :",r))
   mtext(side=1, line=3, outer=T,
    encode("When Z =",z[j]," PDF =",pd[j]))
           }
  rm(zinc,pd,zz,hi,totpdf)
})
END
```

```
MACRO bvu1tran(
 it/?PROMPT(Number of intervals you want to divide the Z range : )/)
chapter
({
#  This is a Macro to transform bivariate uniform
#  variables with the form of z = x + y
#
   ?MESSAGE()
   ?MESSAGE(Range of Z will be from 0 to 2.)
   ?MESSAGE()
   ?T(i)_it
   ?T(z)_seq(0, 2 ,len=(?T(i)+1) )
#
#  Call macro to plot the z=x+y (3 dimensional plot) and
#  the range of z in terms of x and y dimension.
#
   ?bvu1plot1(?T(i),?T(z))
#
#  Compute the volume under the given area by calling the imsl
#  library for double integration via S extention function.
#
  ?T(cd)_bvunif(?T(i),?T(z))
#
#  Call macro to plot the estimated pdf for transformed variable z.
#
   ?bvu1plot3(?T(cd),?T(z),?T(i))
   rm(?T(z),?T(i),?T(cd))
})
END
```

```
MACRO bvu2tran(
it/?PROMPT(Number of intervals you want to divide the Z range : )/)
chapter
({
#  This is a Macro to transform bivariate uniform
#  variables with the form of z = x^2 + y^2
#
   ?MESSAGE()
   ?MESSAGE(Range of Z will be from 0 to 2.)
   ?MESSAGE(Choose range of Z and number of intervals.)
   ?MESSAGE()
   ?T(i)_it
   ?T(z)_seq(0, 2 ,len=(?T(i)+1) )
   ?T(zs)_sqrt(?T(z))
#
#  Call macro to plot the z=x^2+y^2 (3 dimensional plot) and
#  the range of z in terms of x and y dimension.
#
   ?bvu2plot1(?T(i),?T(z),?T(zs))
#
#  Compute the volume under the given area by calling the imsl
#  library for double integration via S extention function.
#
  ?T(cd)_bvunif2(?T(i),?T(z))
#
#  Call macro to plot the estimated pdf for transformed variable z.
#
   ?bvu2plot3(?T(cd),?T(z),?T(zs),?T(i))
   rm(?T(z),?T(zs),?T(i),?T(cd))
})
END
```

```
MACRO bvu1plot1( it,z)
#
# This macro is to plot relationship between z and x,y and
# the range of z in terms of x and y.
#
({
  x_seq(0,1,len=11)
  y_x
  xx_rep(x,rep(11,11))
  yy_rep(y,11)
  xx_matrix(xx,11,11,byrow=T)
  yy_matrix(yy,11,11,byrow=T)
  zz_xx+yy
  mz_max(zz)
  par(mfrow=c(1,2), oma=c(4,0,4,0))
#
# Plot the z in 3 dimensional plane.
#
  persp(zz/mz)
  title(sub="Y(L) & X(R) : [ 0 1 ]")
#
# Plot the x-y axis.
#
  plot(c(-1,2),c(-1,2),type="n", xlab="X", ylab="Y")
  title(sub="Bivariate Uniform ( 0 1 )")
#
# Plot the line x=0 & y=0.
#
  abline(v=0)
  abline(h=0)
#
# Plot the range of z in x-y plane.
#
  abline(z[it+1],-1)
#
# Shade the area representing the defined uniform
# distribution range.
#
  hatch(c(0,0,1,1),c(0,1,1,0), border=T)
  mtext(side=3, line=1, outer=T,
    "Plot of Z = X + Y  and  Range of Transform")
  mtext(side=1, line=1, outer=T,
    "Starting Z : 0  Ending Z : 2")
  rm(x,y,xx,yy,zz,mz)
})
END
```

```
MACRO bvu1plot3(cd,z,i)
#
# This macro is to plot transformed pdf and the original
# pdf contour.
#
({
  par(mfrow=c(1,2), oma=c(5,3,4,0) )
  zinc_z[2]-z[1]
  td_c(0,cd[1:(i-1)])
  cd_cd-td
  pd_cd/zinc
  zz_rep(z,rep(2,i+1))
  hi_rep(pd,rep(2,i))
  hi_c(0,hi,0)
  totpdf_sum(cd)
#
# Plot the transformed pdf.
#
  plot(z[1:i],pd, type="h", xlab="Z", ylab="",
       ylim=c(0,max(hi)), xlim=c(z[1],z[i+1]),
       main=encode("Total PDF =",totpdf),
       sub=encode("Increment of Z :",zinc), err=-1)
  lines(zz,hi)
  mtext(side=2, line=1, outer=T, "PDF")
#
# Plot the range of z in x-y plane.
#
  plot(c(0,0,1,1,0),c(0,1,1,0,0),type="l",
   xlim=c(-1,2), ylim=c(-1,2), xlab="X", ylab="Y")
#
# Plot the line representing range of z.
#
  for(j in 1:(i+1)) {
    abline(z[j],-1)
    }
  title(sub=encode("Starting Z:",z[1],"  Ending Z:",z[i+1]))
  mtext(side=3, line=1, outer=T,
    "Transform of Bivariate Uniform variable with Z = X + Y")
  mtext(side=1, line=1, outer=T,
    "Bivariate Uniform ( 0 1 )" )
  mtext(side=1, line=3, outer=T,
    "This is the overall picture for transformation. If you want to see
     the picture one by one, hit Return at every time GO? appears")
#
# Iterative plot
#
  for( j in 1:i ) {
```

```
#
# Plot the cumulative transformed pdf.
#
  par(mfrow=c(1,2), oma=c(5,3,4,0))
   totpdf_sum(cd[1:j])
   plot(c(z[1],z[2:(j+1)]),c(pd[1],pd[1:j]), type="h", xlab="Z",
        ylab="", ylim=c(0,max(pd[1:j])), xlim=c(z[1],z[j+1]),
        main=encode("Total PDF =",totpdf),
        sub=encode("Increment of Z :", zinc), err=-1)
   lines(zz[1:(1+2*j)], hi[1:(1+2*j)])
  mtext(side=2, line=1, outer=T, "PDF")
#
# Shade current pdf.
#
   hatch(c(z[j],z[j],z[j+1],z[j+1]),
         c(0,pd[j],pd[j],0))
#
# Plot the area representing defined uniform distribution.
#
   plot(c(0,0,1,1,0),c(0,1,1,0,0),type="l",
        xlab="X", ylab="Y")
#
# Plot the range of z
#
    for(k in 1:(j+1)) {
      abline(z[k],-1)
      }
#
# Shade the current area.
#
   hatch(c(max(0,z[j]-1),max(0,z[j+1]-1),min(z[j+1],1),min(z[j],1)),
         c(min(1,z[j]),min(1,z[j+1]),max(0,z[j+1]-1),max(0,z[j]-1)),
         border=F)
   title(sub=encode("Starting Z:",z[j],"  Ending Z:",z[j+1]))
   mtext(side=3, line=1, outer=T,
    "Transform of Bivariate Uniform variable with Z = X + Y")
   mtext(side=1, line=1, outer=T,
    "Bivariate Uniform ( 0 1 )")
   mtext(side=1, line=3, outer=T,
    encode("When Z =",z[j],"  PDF =",pd[j]))
         }
  rm(zinc,pd,td,zz,hi,totpdf)
})
END
```

```
MACRO bvu2plot1( it,z,zs)
#
# This macro is to plot relationship between z and x,y and
# the range of z in terms of x and y.
#
({
  x_seq(0,1,len=11)
  y_x
  xx_rep(x,rep(11,11))
  yy_rep(y,11)
  xx_matrix(xx,11,11,byrow=T)
  yy_matrix(yy,11,11,byrow=T)
  zz_xx^2+yy^2
  mz_max(zz)
  par(mfrow=c(1,2), oma=c(4,0,4,0))
#
# Plot the z in 3 dimensional plane.
#
  persp(zz/mz)
  title(sub="Y(L) & X(R) : [ 0 1 ]")
#
# Plot the x-y axis.
#
  plot(c(-2,2),c(-2,2),type="n", xlab="X", ylab="Y")
  title(sub="Bivariate Uniform ( 0 1 )")
#
# Plot the line representing x=0 & y=0.
#
  abline(v=0)
  abline(h=0)
  cx_rep(0,it+1)
#
# Plot the circle representing range of z.
#
  symbols(cx,cx,circle=zs, add=T, inches=F, err=-1)
  hatch(c(0,0,1,1), c(0,1,1,0))
  mtext(side=3, line=1, outer=T,
    "Plot of Z = X^2 + Y^2  and  Range of Transform")
  mtext(side=1, line=1, outer=T,
    "Starting Z : 0  Ending Z : 2")
  rm(x,y,xx,yy,zz,mz,cx)
})
END
```

```
MACRO bvu2plot3(cd,z,zs,i)
#
# This macro is to plot transformed pdf and the original
# pdf contour.
#
({
  par(mfrow=c(1,2), oma=c(5,3,4,0) )
  zinc_z[2]-z[1]
  td_c(0,cd[1:(i-1)])
  cd_cd-td
  pd_cd/zinc
  zz_rep(z,rep(2,i+1))
  hi_rep(pd,rep(2,i))
  hi_c(0,hi,0)
  totpdf_sum(cd)
#
# Plot the transformed pdf.
#
  plot(z[1:i],pd, type="h", xlab="Z", ylab="",
       ylim=c(0,max(hi)), xlim=c(z[1],z[i+1]),
       main=encode("Total PDF =",totpdf),
       sub=encode("Increment of Z :",zinc), err=-1)
  lines(zz,hi)
  mtext(side=2, line=1, outer=T, "PDF")
#
# Plot the area representing defined uniform distribution
# range.
#
  plot(c(0,0,1,1,0),c(0,1,1,0,0),type="l",
   xlim=c(-2,2), ylim=c(-2,2), xlab="X", ylab="Y")
#
# Plot the circle representing current range of z.
#
  symbols(c(0,0),c(0,0),circle=c(zs[1],zs[i+1]),
          add=T, inches=F, err=-1)
  title(sub=encode("Starting Z:",z[1],"  Ending Z:",z[i+1]))
  mtext(side=3, line=1, outer=T,
    "Transform of Bivariate Uniform variable with Z = X^2 + Y^2")
  mtext(side=1, line=1, outer=T,
    "Bivariate Uniform ( 0 1 )" )
  mtext(side=1, line=3, outer=T,
    "This is the overall picture for transformation. If you want to see
     the picture one by one, hit Return at every time GO? appears")
#
# Iterative plot
#
  for( j in 1:i ) {
```

```
#
# Plot the cumulative transformed pdf.
#
  par(mfrow=c(1,2), oma=c(5,3,4,0))
   totpdf_sum(cd[1:j])
   plot(c(z[1],z[2:(j+1)]),c(pd[1],pd[1:j]), type="h", xlab="Z",
        ylab="", ylim=c(0,max(pd[1:j])), xlim=c(z[1],z[j+1]),
        main=encode("Total PDF =",totpdf),
        sub=encode("Increment of Z :", zinc), err=-1)
   lines(zz[1:(1+2*j)], hi[1:(1+2*j)])
  mtext(side=2, line=1, outer=T, "PDF")
#
# Shade the current transformed pdf.
#
   hatch(c(z[j],z[j],z[j+1],z[j+1]),
         c(0,pd[j],pd[j],0))
#
# Plot the area representing defined uniform distribution
# range.
#
   plot(c(0,0,1,1,0),c(0,1,1,0,0),type="l",
   xlim=c(-2,2), ylim=c(-2,2), xlab="X", ylab="Y")
#
# Plot the circle representing current range of z.
#
   symbols(c(0,0),c(0,0),circle=c(zs[j],zs[j+1]), add=T, inches=F,
           err=-1)
   title(sub=encode("Starting Z:",z[j],"  Ending Z:",z[j+1]))
   mtext(side=3, line=1, outer=T,
    "Transform of Bivariate Uniform variable with Z = X^2 + Y^2")
   mtext(side=1, line=1, outer=T,
    "Bivariate Uniform ( 0 1 )")
   mtext(side=1, line=3, outer=T,
    encode("When Z =",z[j],"  PDF =",pd[j]))
          }
  rm(zinc,pd,td,zz,hi,totpdf)
})
END
```

```
MACRO bvcplot1(x,y,dfx,dfy,dfr)
#
# This macro is to plot relationship between z and x,y and
# the range of z in terms of x and y.
#
({
  xx_rep(x,rep(10,10))
  yy_rep(y,10)
  xx_matrix(xx,10,10,byrow=T)
  yy_matrix(yy,10,10,byrow=T)
  zz_xx/yy
  mz_max(zz)
#
# Plot the z in 3 dimensional plane.
#
  par(mfrow=c(1,2), oma=c(4,0,4,0))
  persp(zz/mz)
  title(sub="Y(L) & X(R) :  [ 0.0001  2 ]")
#
# Plot x-y axis.
#
  plot(x,y,type="n", axes=F, xlab="X", ylab="Y")
  axis(1) ;  axis(2)
# title(sub=encode("Given DFx :",dfx,"  DFy :",dfy))
# Plot the range of z in x-y plane.
#
  abline(0,1/(7*dfr))
  abline(v=0)
#
# Shade the range of z in x-y plane.
#
  hatch(c(0,0,2,2),c(0,2,2,2/(7*dfr)),border=F)
  mtext(side=3, line=1, outer=T,
    "Plot of Z = (X/DFx) / (Y/DFy)  and  Range of Transform")
  mtext(side=1, line=1, outer=T,
    encode("Starting Z:",0,"  Ending Z:",7))
  rm(xx,yy,zz,mz)
})
END
```

```
MACRO bvcplot2(x,y,w,dfx,dfy,dfr)
#
# This macro is to plot the pdf contour and the pdf plane
# in 3 dimensional plane.
#
({
  par(mfrow=c(1,2), oma=c(4,0,4,0))
  mw_max(w)
#
# Plot the pdf contour in x-y plane.
#
  contour(x,y,w, axes=F, xlab="X", ylab="Y")
  axis(1) ;  axis(2)
#
# Plot the range of z n x-y plane.
#
  abline(v=0)
  abline(0,1/(7*dfr))
#
# Shade the range of z.
#
  hatch(c(0,0,2,2),c(0,2,2,2/(7*dfr)), border=F)
  title(sub="Starting Z :  0    Ending Z :  7")
#
# Plot the pdf plane in 3 dimensional plane.
#
  persp(w/mw)
  title(sub="Y(L) & X(R) :  [ 0.0001  2 ]")
  mtext(side=3, line=1, outer=T,
    "Bivariate Chisquare PDF Surface")
  mtext(side=1, line=1, outer=T,
    encode("Given DFx :",dfx,"  DFy :",dfy))
  rm(mw)
})
END
```

```
MACRO bvcplot3(cd,z,i,x,y,w,dfx,dfy,dfr)
#
# This macro is to plot transformed pdf and the original
# pdf contour.
#
({
  par(mfrow=c(1,2), oma=c(5,3,4,0) )
  zinc_z[2]-z[1]
  pd_cd/zinc
  totpdf_sum(cd)
  zz_rep(z,rep(2,(i+1)))
  hi_rep(pd,rep(2,i))
  hi_c(0,hi,0)
#
# Plot the transformed pdf
#
  plot(z[1:i],pd, type="h", xlab="Z", ylab="",
       ylim=c(0,max(hi)), xlim=c(z[1],z[i+1]),
       main=encode("Total PDF =",totpdf),
       sub=encode("Increment of Z :",zinc), err=-1)
  lines(zz,hi)
  mtext(side=2, line=1, outer=T, "PDF")
#
# Plot the pdf contour in x-y plane.
#
  contour(x,y,w, axes=F, xlab="X", ylab="Y" )
  axis(1) ;  axis(2)
#
# Plot the range of z on the contour plot.
#
  for ( j in 2:(i+1) ) {
    abline(0,1/(z[j]*dfr))
    }
  title(sub="Starting Z : 0     Ending Z : 7")
  mtext(side=3, line=1, outer=T,
    "Transform of Bivariate Chisquare variable
      with Z = (X/DFx) / (Y/DFy)")
  mtext(side=1, line=1, outer=T,
    encode("Given DFx :",dfx,"  DFy :",dfy))
  mtext(side=1, line=3, outer=T,
    "This is the overall picture for transformation. If you want to see
     the picture one by one, hit Return at every time GO? appears")
#
# Iterative Plot
#
  for( j in 1:i ) {
  par(mfrow=c(1,2), oma=c(5,3,4,0))
  totpdf_sum(cd[1:j])
```

```
#
# Plot the cumulative transformed pdf.
#
  plot(c(z[1],z[2:(j+1)]),c(pd[1],pd[1:j]), type="h", xlab="Z",
        ylab="", ylim=c(0,max(pd[1:j])), xlim=c(z[1],z[j+1]),
        main=encode("Total PDF =",totpdf),
        sub=encode("Increment of Z :", zinc), err=-1)
  lines(zz[1:(1+2*j)], hi[1:(1+2*j)])
  mtext(side=2, line=1, outer=T, "PDF")
#
# Shade current pdf
#
  hatch(c(z[j],z[j],z[j+1],z[j+1]),
        c(0,pd[j],pd[j],0))
#
# Plot the pdf contour.
#
  contour(x,y,w, axes=F, xlab="X", ylab="Y")
  axis(1) ;  axis(2)
  title(sub=encode("Starting Z:",z[j]," Ending Z:",z[j+1]))
#
# Plot the cumulative range of z.
#
  for ( k in 1:j ) {
    abline(0,1/(z[k+1]*dfr))
    }
#
# Compute the coordinates for current z range and
# shade the current z range.
#
  x1_min(2,2*z[j]*dfr)
  x2_min(2,2*z[j+1]*dfr)
  y1_min(2,2/max(.000001,z[j]*dfr))
  y2_min(2,2/(z[j+1]*dfr))
  hatch(c(0,x1,x2), c(0,y1,y2),border=F, angle=135)
  mtext(side=3, line=1, outer=T,
    "Transform of Bivariate Chisquare variable
        with Z = (X/DFx) / (Y/DFy)")
  mtext(side=1, line=1, outer=T,
    encode("DFx :",dfx,"  DFy :",dfy))
  mtext(side=1, line=3, outer=T,
    encode("When Z =",z[j],"    PDF =",pd[j]))
    }
  rm(zinc,pd,zz,hi,x1,x2,y1,y2,totpdf)
})
END
```

```
MACRO bvncplot1(x,y,df,my)
#
# This macro is to plot relationship between z and x,y and
# the range of z in terms of x and y.
#
({
  xx_rep(x,rep(11,11))
  yy_rep(y,11)
  xx_matrix(xx,11,11,byrow=T)
  yy_matrix(yy,11,11,byrow=T)
  zz_xx/yy
  mz_max(zz)
  par(mfrow=c(1,2), oma=c(6,0,2,0))
#
# Plot z in 3 dimensional plane.
#
  persp(zz/mz)
  title(sub="Y[Left]  X[Right]")
#
# Plot the x-y axis.
#
  plot(x,y,type="n", axes=F, xlab="X", ylab="Y",
       ylim=c(0,my))
  axis(1) ;  axis(2)
#
# Plot the range of z in x-y plane.
#
  abline(0,-1/4)
  abline(0, 1/4)
#
# Shade the z range
#
  hatch(c(0,-3,-3,3,3), c(0,3/4,my,my,3/4), border=F)
  mtext(side=3, line=0, outer=T,
    "Plot of T = X / Y  and  Range of Transform")
  mtext(side=1, line=1, outer=T,
    encode("X = N(0,1)  Y = SQRT( Chisquare[ DF:",df,"] /",df,
    ")"))
  mtext(side=1, line=3, outer=T,
    encode("X : [ -3  3 ]   Y : [ 0",y[11],"]"))
  mtext(side=1, line=5, outer=T,
    "Starting T : -4,  Ending T : 4")
  rm(xx,yy,zz,mz)
})
END
```

```
MACRO bvncplot2(x,y,w,df,my)
#
# This macro is to plot the pdf contour and the pdf plane
# in 3 dimensional plane.
#
({
  par(mfrow=c(1,2), oma=c(3,0,2,0))
  mw_max(w)
#
# Plot the pdf contour
#
  contour(x,y,w, axes=F, ylim=c(0,my),
          xlab="X", ylab="Y")
  axis(1) ;  axis(2)
#
# Plot the range of z in x-y plane.
#
  abline(0,-1/4)
  abline(0, 1/4)
#
# Shade the z range.
#
  hatch(c(0,-3,-3,3,3), c(0,3/4,my,my,3/4), border=F)
  title(sub="Starting T: -4  Ending T:  4")
#
# Plot the pdf plane in 3 dimensional plane.
#
  persp(w/mw)
  title(sub="Y[Left]  X[Right]")
  mtext(side=3, line=0, outer=T,
    encode("PDF Surface for N(0,1) & Chisq[ DF:",df,"]"))
  mtext(side=1, line=1, outer=T,
    encode("X = N(0,1)  Y = SQRT( Chisquare[ DF:",df,"] / ,df,
     ")"))
  rm(mw)
})
END
```

```
MACRO bvncplot3(cd,z,i,x,y,w,df,my)
#
# This macro is to plot transformed pdf and the original
# pdf contour.
#
({
  par(mfrow=c(1,2), oma=c(5,3,4,0) )
  zinc_z[2]-z[1]
  pd_cd/zinc
  totpdf_sum(cd)
  zz_rep(z,rep(2,(i+1)))
  hi_rep(pd,rep(2,i))
  hi_c(0,hi,0)
#
# Plot the transformed pdf.
#
  plot(z[1:i],pd, type="h", xlab="T", ylab="",
       ylim=c(0,max(hi)), xlim=c(z[1],z[i+1]),
       main=encode("Total PDF =",totpdf),
       sub=encode("Increment of T :",zinc), err=-1)
  lines(zz,hi)
  mtext(side=2, line=1, outer=T, "PDF")
#
# Plot the pdf contour in x-y plane.
#
  contour(x,y,w, axes=F, ylim=c(0,my),
          xlab="X", ylab="Y")
  axis(1) ;  axis(2)
#
# To avoid zero denominator, if z value is zero, the z will
# be replaced by .00000000001.
#
  zp_z
  zp_ifelse(zp!=0, zp, .00000000001)
#
# Plot the range of z on the contour plot.
#
  for ( j in 1:(i+1) ) {
    abline(0,1/zp[j])
    }
  title(sub="Starting T : -4    Ending T : 4")
  mtext(side=3, line=1, outer=T,
    "Transform of Normal(X) & Chisquare variable(Yp)
         with T = X/SQRT(Yp/DFy)")
  mtext(side=1, line=1, outer=T,
    encode("X : Normal(0,1),  Y : SQRT(Yp/DF), where Yp : Chisquare with
DF",
    df))
  mtext(side=1, line=3, outer=T,
    "This is the overall picture for transformation. If you want to see
     the picture one by one, hit Return at every time GO? appears")
```

```
#
# Iterative plot
#
  for( j in 1:i ) {
  par(mfrow=c(1,2), oma=c(5,3,5,0))
  totpdf_sum(cd[1:j])
#
# Plot the cumulative transformed pdf
#
  plot(c(z[1],z[2:(j+1)]),c(pd[1],pd[1:j]), type="h", xlab="T",
        ylab="", ylim=c(0,max(pd[1:j])), xlim=c(z[1],z[j+1]),
        main=encode("Total PDF =",totpdf),
        sub=encode("Increment of Z :", zinc), err=-1)
  lines(zz[1:(1+2*j)], hi[1:(1+2*j)])
  mtext(side=2, line=1, outer=T, "PDF")
#
# Shade the current pdf
#
  hatch(c(z[j],z[j],z[j+1],z[j+1]),
        c(0,pd[j],pd[j],0))
#
# Plot the pdf contour.
#
  contour(x,y,w, axes=F, ylim=c(0,my),
          xlab="X", ylab="Y")
  axis(1) ;  axis(2)
  title(sub=encode("Starting T:",z[j]," Ending T:",z[j+1]))
#
# Plot the cumulative z range on the contour plot.
#
  for ( k in 1:(j+1) ) {
    abline(0,1/zp[k])
    }
#
# Compute the coordinates for current z range and
# shade the current area.
#
  mx1_zp[j]*my
  mx1_min(3, max(-3, mx1))
  mx2_zp[j+1]*my
  mx2_min(3, max(-3, mx2))
  my1_abs(3/zp[j])
  my1_min(my1, my)
  my2_abs(3/zp[j+1])
  my2_min(my2, my)
  hatch(c(0,mx1,mx2), c(0,my1,my2), border=F)
  mtext(side=3, line=1, outer=T,
    "Transform of Normal(X) & Chisquare variable(Yp)
           with T = X/SQRT(Yp/DFy)")
```

```
  mtext(side=1, line=1, outer=T,
    encode("X : Normal(0,1),  Y : SQRT(Yp/DF), where Yp :
            Chisquare with DF", df))
  mtext(side=1, line=3, outer=T,
    encode("When T =",z[j],"    PDF =",pd[j]))
    }
  rm(zinc,pd,zz,zp,hi,totpdf)
})
END
```

Appendix D: Macros for Hypothesis Testing

Table of Contents

```
MACRO hypo
({
   ?MESSAGE()
   item_c("Test of Mean with Known Sigma",
          "Test of Mean with Unknown Sigma")
    action_c("?test1", "?test2")
    menu( item, action)
    rm( item, action)
})
END
```

```
MACRO test1
({
  ?MESSAGE()
  ?MESSAGE(Null Hypothesis  Ho :  MU = MUo)
  ?MESSAGE(Please select proper alternative hypothesis.)
  ?MESSAGE()
  item_c(" Ha :  MU  >  MUo  (Right tail test)",
         " Ha :  MU  <  MUo  (Left  tail test)",
         " Ha :  MU !=  MUo  (Two   tail test)")
  action_c("?test1r",
           "?test1l",
           "?test1t")
  menu( item, action)
  rm( item, action)
})
END
```

```
MACRO test1r(
xb/?PROMPT(X bar       : )/,
nv/?PROMPT(Sample size : )/,
sv/?PROMPT(Sigma       : )/,
m0/?PROMPT(MUo         : )/,
al/?PROMPT(Alpha       : )/,
es/?PROMPT(Effect Size for computing power : )/)
({
#
# This macro is to test for a mean for a normal population
# with known sigma.
# ( Right tail test )
#
   ?MESSAGE()
   ?T(x)_xb
   ?T(n)_nv
   ?T(s)_sv
   ?T(m0)_m0
   ?T(a)_al
   ?MESSAGE()
   ?MESSAGE(Effect size is the distance between MUo and Mu1,)
   ?MESSAGE(and it is absolute value.)
   ?MESSAGE()
   ?T(es)_es
   ?T(ss)_?T(s)/sqrt(?T(n))
   ?T(m1)_?T(m0)+?T(es)
#
# Compute quantile for following populations from cdf=.001 to .999;
#    normal population : N(MUo, sigma)  & N(MUo, sigma/sqrt(n))
#                        N(MU1, sigma)  & N(MU1, sigma/sqrt(n))
#
   x1_qnorm(.001, ?T(m0), ?T(s))
   x2_qnorm(.999, ?T(m0), ?T(s))
   x3_qnorm(.001, ?T(m1), ?T(s))
   x4_qnorm(.999, ?T(m1), ?T(s))
   x5_qnorm(.001, ?T(m0), ?T(ss))
   x6_qnorm(.999, ?T(m0), ?T(ss))
   x7_qnorm(.001, ?T(m1), ?T(ss))
   x8_qnorm(.999, ?T(m1), ?T(ss))
   ?T(x1)_seq(x1,x2, len=100)
   ?T(x2)_seq(x3,x4, len=100)
   ?T(x3)_seq(x5,x6, len=100)
   ?T(x4)_seq(x7,x8, len=100)
   ?T(x12)_max(?T(x1),?T(x2))
   ?T(n12)_min(?T(x1),?T(x2))
   ?T(x34)_max(?T(x3),?T(x4))
   ?T(n34)_min(?T(x3),?T(x4))
#
# Compute PDF for normal population.
#
   ?T(pm0)_dnorm(?T(x1), ?T(m0), ?T(s))
   ?T(pm1)_dnorm(?T(x2), ?T(m1), ?T(s))
```

```
# Compute maximum pdf for both normal distribution
#
   ?T(mx0)_dnorm(?T(m0), ?T(m0), ?T(s))
   ?T(mx1)_dnorm(?T(m1), ?T(m1), ?T(s))
#
# Make proper label
#
   label1_encode("MUo =",?T(m0)," MU1 =", ?T(m1),
          " Known sigma =",?T(s)," Effect Size =",?T(es))
#
# Call macro to plot the screen 1.
#
   ?screen1(?T(m0),?T(m1),?T(x1),?T(x2),?T(pm0),?T(pm1),
            ?T(mx0),?T(mx1),?T(x12),?T(n12),label1)
#
# Compute Xbar critical value.
#
   ?T(xbc)_qnorm((1-?T(a)), ?T(m0), ?T(ss))
   ?T(xc0)_dnorm(?T(xbc), ?T(m0), ?T(ss))
   ?T(xc1)_dnorm(?T(xbc), ?T(m1), ?T(ss))
#
# Compute PDF for sampling distribution.
#
   ?T(px0)_dnorm(?T(x3), ?T(m0), ?T(ss))
   ?T(px1)_dnorm(?T(x4), ?T(m1), ?T(ss))
#
# Compute maximum pdf for both sampling distribution
#
   ?T(xx0)_dnorm(?T(m0), ?T(m0), ?T(ss))
   ?T(xx1)_dnorm(?T(m1), ?T(m1), ?T(ss))
#
# Compute Likelihood ratio
#
   ?T(xx)_seq(?T(n34),?T(x34), len=100)
   ?T(lr)_exp((-?T(n)*((?T(xx)-?T(m0))^2))/(2*(?T(s)^2)))
#
# Compute the likelihood ratio for X critical.
#
   ?T(k)_exp((-?T(n)*((?T(xbc)-?T(m0))^2))/(2*(?T(s)^2)))
#
# Compute Beta (= 1- power)
#
   ?T(b)_pnorm(?T(xbc), ?T(m1), ?T(ss))
#
# Make proper label.
#
  label1_encode("MUo =",?T(m0)," MU1 =",?T(m1)," Known Sigma =",
                ?T(s))
  label2_encode("Effect size =",?T(es)," Sample Size =",?T(n))
  label3_encode("//// : Alpha =",?T(a),
          "     \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ : Beta =", ?T(b))
```

```
  label4_encode("Xbar crit =",?T(xbc),"  LR =",?T(k))
  label_c(label1, label2, label3, label4)
#
# Call macro to plot the screen 2.
#
   ?screen2r(?T(m0),?T(m1),?T(x3),?T(x4),?T(px0),?T(px1),?T(xx0),
          ?T(xx1),?T(xx),?T(lr),?T(k),?T(xbc),?T(xc0),?T(xc1),
          ?T(x34),?T(n34),x6,x7,label)
#
# Compute PDF in Z scale
#
   ?T(m1p)_(?T(m1)-?T(m0))/?T(ss)
   ?T(zc)_qnorm(1-?T(a))    # Z critical value
   ?T(zc0)_dnorm(?T(zc), 0, 1)
   ?T(zc1)_dnorm(?T(zc), ?T(m1p), 1)
   x9_qnorm(.001, 0, 1)
   x10_qnorm(.999, 0, 1)
   x11_qnorm(.001, ?T(m1p), 1)
   x12_qnorm(.999, ?T(m1p), 1)
   ?T(x5)_seq(x9,x10, len=100)
   ?T(x6)_seq(x11,x12, len=100)
   ?T(pz0)_dnorm(?T(x5), 0, 1)
   ?T(pz1)_dnorm(?T(x6), ?T(m1p), 1)
   ?T(x56)_max(?T(x5),?T(x6))
   ?T(n56)_min(?T(x5),?T(x6))
#
# Compute Power Function
#
   ?T(px)_seq(?T(m0)-2*?T(s), ?T(m0)+2*?T(s), len=100)
   ?T(mup)_(?T(m0)-?T(px))/?T(ss)
   ?T(pwr)_1-pnorm(?T(zc)+?T(mup))
   ?T(pw1)_1-pnorm(?T(zc)-?T(m1p))
#
# Compute maximum pdf for both z distribution
#
   ?T(zx0)_dnorm(0, 0, 1)
   ?T(zx1)_dnorm(?T(m1p), ?T(m1p), 1)
#
# Make proper label
#
   label5_encode("When MU = ",?T(m1),"   Power = ",?T(pw1))
   label_c(label, label5)
#
# Call macro to plot the screen 3.
#
   ?screen3r(?T(m1),?T(m1p),?T(x5),?T(x6),?T(pz0),?T(pz1),?T(zx0),
          ?T(zx1),?T(px),?T(zc),?T(pwr),?T(zc0),?T(zc1),
          ?T(pw1),?T(x56),?T(n56),x10,x11,label)
```

```
#
# Test results
#
   ?T(pv)_1-pnorm(?T(x), ?T(m0), ?T(ss))
   ?T(z)_(?T(x)-?T(m0))/?T(ss)
   label6_encode("Xbar* =",?T(x)," Xbar crit =",?T(xbc),
         " Z* =",?T(z)," Z crit =",?T(zc))
   label_c(label,label6)
#
# Call macro to plot the screen 4.
#
   xv1_rbind(?T(x3),?T(x4))
   pxv_rbind(?T(px0),?T(px1))
   xxv_c(?T(xx0),?T(xx1))
   xcv_c(?T(xc0),?T(xc1))
   xn1_c(?T(x34),?T(n34))
   xp1_c(x6,x7)
   xv2_rbind(?T(x5),?T(x6))
   pzv_rbind(?T(pz0),?T(pz1))
   zxv_c(?T(zx0),?T(zx1))
   zcv_c(?T(zc0),?T(zc1))
   xn2_c(?T(x56),?T(n56))
   xp2_c(x10,x11)
   mmv_c(?T(m0),?T(m1),?T(m1p))
   ?screen4r(?T(x),?T(xbc),?T(pv),?T(zc),?T(z),
            mmv,xv1,pxv,xxv,xcv,xn1,xp1,xv2,pzv,
            zxv,zcv,xn2,xp2,label)
   rm(?T(x),?T(n),?T(s),?T(m0),?T(m1),?T(a),?T(es),?T(ss),
      ?T(x1),?T(x2),?T(x3),?T(x4),?T(x5),?T(x6),?T(x12),?T(n12),
      ?T(x34),?T(n34),?T(x56),?T(n56),?T(pm0),?T(pm1),?T(mx0),?T(mx1),
      ?T(xbc),?T(xc0),?T(xc1),?T(px0),?T(px1),?T(xx0),?T(xx1),?T(xx),
      ?T(lr),?T(k),?T(m1p),?T(pz0),?T(pz1),?T(px),?T(mup),?T(pwr),
      ?T(pw1),?T(zx0),?T(zx1),?T(pv),?T(zc),?T(zc0),?T(zc1),
      x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,label1,label2,label3,
      label4,label5,label6,xv1,xv2,pxv,pzv,xxv,zxv,xcv,zcv,
      xn1,xn2,mmv,xp1,xp2,label,?T(b),?T(z))
})
END
```

```
MACRO test11(
xb/?PROMPT(X bar       : )/,
nv/?PROMPT(Sample size : )/,
sv/?PROMPT(Sigma       : )/,
m0/?PROMPT(MUo         : )/,
al/?PROMPT(Alpha       : )/,
es/?PROMPT(Effect Size for computing power : )/)
({
#
# This macro is to test for a mean for a normal population
# with known sigma.
# ( Left tail test )
#
   ?MESSAGE()
   ?T(x)_xb
   ?T(n)_nv
   ?T(s)_sv
   ?T(m0)_m0
   ?T(a)_al
   ?MESSAGE()
   ?MESSAGE(Effect size is the distance between MUo and Mu1,)
   ?MESSAGE(and it is absolute value.)
   ?MESSAGE()
   ?T(es)_es
   ?T(ss)_?T(s)/sqrt(?T(n))
   ?T(m1)_?T(m0)-?T(es)
#
# Compute quantile for following populations from cdf=.001 to .999;
#    normal population : N(MUo, sigma)  & N(MUo, sigma/sqrt(n))
#                        N(MU1, sigma)  & N(MU1, sigma/sqrt(n))
#
   x1_qnorm(.001, ?T(m0), ?T(s))
   x2_qnorm(.999, ?T(m0), ?T(s))
   x3_qnorm(.001, ?T(m1), ?T(s))
   x4_qnorm(.999, ?T(m1), ?T(s))
   x5_qnorm(.001, ?T(m0), ?T(ss))
   x6_qnorm(.999, ?T(m0), ?T(ss))
   x7_qnorm(.001, ?T(m1), ?T(ss))
   x8_qnorm(.999, ?T(m1), ?T(ss))
   ?T(x1)_seq(x1,x2, len=100)
   ?T(x2)_seq(x3,x4, len=100)
   ?T(x3)_seq(x5,x6, len=100)
   ?T(x4)_seq(x7,x8, len=100)
   ?T(x12)_max(?T(x1),?T(x2))
   ?T(n12)_min(?T(x1),?T(x2))
   ?T(x34)_max(?T(x3),?T(x4))
   ?T(n34)_min(?T(x3),?T(x4))
#
# Compute PDF for normal population.
#
   ?T(pm0)_dnorm(?T(x1), ?T(m0), ?T(s))
   ?T(pm1)_dnorm(?T(x2), ?T(m1), ?T(s))
```

```
#
# Compute maximum pdf for both normal distribution
#
   ?T(mx0)_dnorm(?T(m0), ?T(m0), ?T(s))
   ?T(mx1)_dnorm(?T(m1), ?T(m1), ?T(s))
#
# Make proper label
#
   label1_encode("  MUo =",?T(m0),"  MU1 =", ?T(m1),
         "  Known sigma =",?T(s),"  Effect Size = ",?T(es))
#
# Call macro to plot the screen 1.
#
   ?screen1(?T(m0),?T(m1),?T(x1),?T(x2),?T(pm0),?T(pm1),
            ?T(mx0),?T(mx1),?T(x12),?T(n12),label1)
#
# Compute Xbar critical value.
#
   ?T(xbc)_qnorm(?T(a), ?T(m0), ?T(ss))
   ?T(xc0)_dnorm(?T(xbc), ?T(m0), ?T(ss))
   ?T(xc1)_dnorm(?T(xbc), ?T(m1), ?T(ss))
#
# Compute PDF for sampling distribution.
#
   ?T(px0)_dnorm(?T(x3), ?T(m0), ?T(ss))
   ?T(px1)_dnorm(?T(x4), ?T(m1), ?T(ss))
#
# Compute maximum pdf for both sampling distribution
#
   ?T(xx0)_dnorm(?T(m0), ?T(m0), ?T(ss))
   ?T(xx1)_dnorm(?T(m1), ?T(m1), ?T(ss))
#
# Compute Likelihood ratio
#
   ?T(xx)_seq(?T(n34),?T(x34), len=100)
   ?T(lr)_exp((-?T(n)*((?T(xx)-?T(m0))^2))/(2*(?T(s)^2)))
#
# Compute the likelihood ratio for X critical.
#
   ?T(k)_exp((-?T(n)*((?T(xbc)-?T(m0))^2))/(2*(?T(s)^2)))
#
# Compute Beta (= 1- power)
#
   ?T(b)_1-pnorm(?T(xbc), ?T(m1), ?T(ss))
#
# Make proper label.
#
  label1_encode("MUo =",?T(m0),"  MU1 =",?T(m1),"  Known Sigma =",
             ?T(s))
  label2_encode("Effect size =",?T(es),"  Sample Size =",?T(n))
  label3_encode("//// : Alpha =",?T(a),"    \\\\\\\\\\\\\\\\ :  Beta =",
             ?T(b))
```

```
  label4_encode("Xbar crit =",?T(xbc),"  LR =",?T(k))
  label_c(label1, label2, label3, label4)
#
# Call macro to plot the screen 2.
#
   ?screen21(?T(m0),?T(m1),?T(x3),?T(x4),?T(px0),?T(px1),?T(xx0),
          ?T(xx1),?T(xx),?T(lr),?T(k),?T(xbc),?T(xc0),?T(xc1),
          ?T(x34),?T(n34),x5,x8,label)
#
# Compute PDF in Z scale
#
   ?T(m1p)_(?T(m1)-?T(m0))/?T(ss)
   ?T(zc)_qnorm(?T(a))    # Z critical value
   ?T(zc0)_dnorm(?T(zc), 0, 1)
   ?T(zc1)_dnorm(?T(zc), ?T(m1p), 1)
   x9_qnorm(.001, 0, 1)
   x10_qnorm(.999, 0, 1)
   x11_qnorm(.001, ?T(m1p), 1)
   x12_qnorm(.999, ?T(m1p), 1)
   ?T(x5)_seq(x9,x10, len=100)
   ?T(x6)_seq(x11,x12, len=100)
   ?T(pz0)_dnorm(?T(x5), 0, 1)
   ?T(pz1)_dnorm(?T(x6), ?T(m1p), 1)
   ?T(x56)_max(?T(x5),?T(x6))
   ?T(n56)_min(?T(x5),?T(x6))
#
# Compute Power Function
#
   ?T(px)_seq(?T(m0)-2*?T(s), ?T(m0)+2*?T(s), len=100)
   ?T(mup)_(?T(m0)-?T(px))/?T(ss)
   ?T(pwr)_pnorm(-?T(zc)+?T(mup))
   ?T(pw1)_pnorm(-?T(zc)-?T(m1p))
#
# Compute maximum pdf for both z distribution
#
   ?T(zx0)_dnorm(0, 0, 1)
   ?T(zx1)_dnorm(?T(m1p), ?T(m1p), 1)
#
# Make proper label
#
   label5_encode("When MU = ",?T(m1),"   Power = ",?T(pw1))
   label_c(label, label5)
#
# Call macro to plot the screen 3.
#
   ?screen31(?T(m1),?T(m1p),?T(x5),?T(x6),?T(pz0),?T(pz1),?T(zx0),
          ?T(zx1),?T(px),?T(zc),?T(pwr),?T(zc0),?T(zc1),
          ?T(pw1),?T(x56),?T(n56),x9,x12,label)
```

```
#
# Test results
#
   ?T(pv)_pnorm(?T(x), ?T(m0), ?T(ss))
   ?T(z)_(?T(x)-?T(m0))/?T(ss)
   label6_encode("Xbar* =",?T(x),"  Xbar crit =",?T(xbc),
        "  Z* =",?T(z),"  Z crit =",?T(zc))
   label_c(label,label6)
#
# Call macro to plot the screen 4.
#
   xv1_rbind(?T(x3),?T(x4))
   pxv_rbind(?T(px0),?T(px1))
   xxv_c(?T(xx0),?T(xx1))
   xcv_c(?T(xc0),?T(xc1))
   xn1_c(?T(x34),?T(n34))
   xp1_c(x5,x8)
   xv2_rbind(?T(x5),?T(x6))
   pzv_rbind(?T(pz0),?T(pz1))
   zxv_c(?T(zx0),?T(zx1))
   zcv_c(?T(zc0),?T(zc1))
   xn2_c(?T(x56),?T(n56))
   xp2_c(x9,x12)
   mmv_c(?T(m0),?T(m1),?T(m1p))
   ?screen41(?T(x),?T(xbc),?T(pv),?T(zc),?T(z),
             mmv,xv1,pxv,xxv,xcv,xn1,xp1,xv2,pzv,
             zxv,zcv,xn2,xp2,label)
   rm(?T(x),?T(n),?T(s),?T(m0),?T(m1),?T(a),?T(es),?T(ss),
      ?T(x1),?T(x2),?T(x3),?T(x4),?T(x5),?T(x6),?T(x12),?T(n12),
      ?T(x34),?T(n34),?T(x56),?T(n56),?T(pm0),?T(pm1),?T(mx0),?T(mx1),
      ?T(xbc),?T(xc0),?T(xc1),?T(px0),?T(px1),?T(xx0),?T(xx1),?T(xx),
      ?T(lr),?T(k),?T(m1p),?T(pz0),?T(pz1),?T(px),?T(mup),?T(pwr),
      ?T(pw1),?T(zx0),?T(zx1),?T(pv),?T(zc),?T(zc0),?T(zc1),
      x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,label1,label2,label3,
      label4,label5,label6, label,xv1,xv2,pxv,pzv,xxv,zxv,xcv,
      zcv,xn1,xn2,mmv,xp1,xp2,?T(b),?T(z))
})
END
```

```
MACRO test1t(
xb/?PROMPT(X bar       : )/,
nv/?PROMPT(Sample size : )/,
sv/?PROMPT(Sigma       : )/,
m0/?PROMPT(MUo         : )/,
al/?PROMPT(Alpha       : )/,
es/?PROMPT(Effect Size for computing power : )/)
({
#
# This macro is to test for a mean for a normal population
# with known sigma.
# ( Two tail test )
#
   ?MESSAGE()
   ?MESSAGE(For two tail test, alternative distribution should)
   ?MESSAGE(exist in both sides of the null distribution, however,)
   ?MESSAGE( for simplification of plotting, only right side )
   ?MESSAGE(alternative distribution will be plotted.)
   ?MESSAGE()
   ?T(x)_xb
   ?T(n)_nv
   ?T(s)_sv
   ?T(m0)_m0
   ?T(a)_al
   ?MESSAGE()
   ?MESSAGE(Effect size is the distance between MUo and Mu1,)
   ?MESSAGE( and it is absolute value.)
   ?MESSAGE()
   ?T(es)_abs(es)
   ?T(ss)_?T(s)/sqrt(?T(n))
   ?T(m1)_?T(m0)+?T(es)
#
# Compute quantile for following populations from cdf=.001 to .999;
#    normal population : N(MUo, sigma)  & N(MUo, sigma/sqrt(n))
#                        N(MU1, sigma)  & N(MU1, sigma/sqrt(n))
#
   x1_qnorm(.001, ?T(m0), ?T(s))
   x2_qnorm(.999, ?T(m0), ?T(s))
   x3_qnorm(.001, ?T(m1), ?T(s))
   x4_qnorm(.999, ?T(m1), ?T(s))
   x5_qnorm(.001, ?T(m0), ?T(ss))
   x6_qnorm(.999, ?T(m0), ?T(ss))
   x7_qnorm(.001, ?T(m1), ?T(ss))
   x8_qnorm(.999, ?T(m1), ?T(ss))
   ?T(x1)_seq(x1,x2, len=100)
   ?T(x2)_seq(x3,x4, len=100)
   ?T(x3)_seq(x5,x6, len=100)
   ?T(x4)_seq(x7,x8, len=100)
   ?T(x12)_max(?T(x1),?T(x2))
   ?T(n12)_min(?T(x1),?T(x2))
   ?T(x34)_max(?T(x3),?T(x4))
   ?T(n34)_min(?T(x3),?T(x4))
```

```
# Compute PDF for normal population.
#
   ?T(pm0)_dnorm(?T(x1), ?T(m0), ?T(s))
   ?T(pm1)_dnorm(?T(x2), ?T(m1), ?T(s))
#
# Compute maximum pdf for both normal distribution
#
   ?T(mx0)_dnorm(?T(m0), ?T(m0), ?T(s))
   ?T(mx1)_dnorm(?T(m1), ?T(m1), ?T(s))
#
# Make proper label
#
   label1_encode("  MUo =",?T(m0),"  MU1 =", ?T(m1),
         "  Known sigma =",?T(s),"  Effect Size = ",?T(es))
#
# Call macro to plot the screen 1.
#
   ?screen1(?T(m0),?T(m1),?T(x1),?T(x2),?T(pm0),?T(pm1),
          ?T(mx0),?T(mx1),?T(x12),?T(n12),label1)
#
# Compute Xbar critical value.
#
   xbcr_qnorm((1-(?T(a)/2)), ?T(m0), ?T(ss))
   xbcl_qnorm((?T(a)/2), ?T(m0), ?T(ss))
   ?T(xbc)_c(xbcr, xbcl)
   xc0r_dnorm(xbcr, ?T(m0), ?T(ss))
   xc0l_dnorm(xbcl, ?T(m0), ?T(ss))
   ?T(xc0)_c(xc0r, xc0l)
   xc1r_dnorm(xbcr, ?T(m1), ?T(ss))
   xc1l_dnorm(xbcl, ?T(m1), ?T(ss))
   ?T(xc1)_c(xc1r, xc1l)
#
# Compute PDF for sampling distribution.
#
   ?T(px0)_dnorm(?T(x3), ?T(m0), ?T(ss))
   ?T(px1)_dnorm(?T(x4), ?T(m1), ?T(ss))
#
# Compute maximum pdf for both sampling distribution
#
   ?T(xx0)_dnorm(?T(m0), ?T(m0), ?T(ss))
   ?T(xx1)_dnorm(?T(m1), ?T(m1), ?T(ss))
#
# Compute Likelihood ratio
#
   ?T(xx)_seq(?T(n34),?T(x34), len=100)
   ?T(lr)_exp((-?T(n)*((?T(xx)-?T(m0))^2))/(2*(?T(s)^2)))
#
# Compute the likelihood ratio for X critical.
#
   kr_exp((-?T(n)*((xbcr-?T(m0))^2))/(2*(?T(s)^2)))
   kl_exp((-?T(n)*((xbcl-?T(m0))^2))/(2*(?T(s)^2)))
   ?T(k)_c(kr, kl)
```

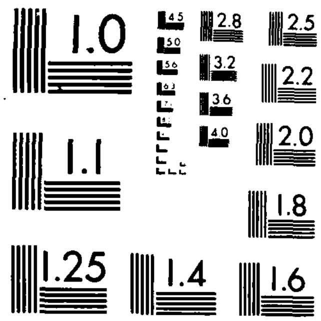
```
# Compute Beta (= 1- power)
#
  ?T(b)_pnorm(xbcr,?T(m1),?T(ss)) - pnorm(xbcl,?T(m1),?T(ss))
#
# Make proper label.
#
 label1_encode("MUo =",?T(m0)," MU1 =",?T(m1)," Known Sigma =",
               ?T(s))
 label2_encode("Effect size =",?T(es)," Sample Size =",?T(n))
 label3_encode("//// : Alpha =",?T(a),"    \\\\\\\\\\\\\\\ : Beta =",
               ?T(b))
 label4_encode("Xbar crit =",?T(xbc)[1],"or",?T(xbc)[2]," LR =",
               ?T(k)[1])
 label_c(label1, label2, label3, label4)
#
# Call macro to plot the screen 2.
#
  ?screen2t(?T(m0),?T(m1),?T(x3),?T(x4),?T(px0),?T(px1),?T(xx0),
           ?T(xx1),?T(xx),?T(lr),?T(k),?T(xbc),?T(xc0),?T(xc1),
           ?T(x34),?T(n34),x5,x6,x7,label)
#
# Compute PDF in Z scale
#
  ?T(m1p)_(?T(m1)-?T(m0))/?T(ss)
  zcr_qnorm((1-?T(a)/2), 0, 1)   # Z critical value (Right side value)
  zcl_qnorm(?T(a)/2, 0, 1)       # Z critical value (Left side value)
  ?T(zc)_c(zcr, zcl)
  ?T(zc0)_dnorm(?T(zc), 0, 1)
  zc1r_dnorm(zcr, ?T(m1p), 1)
  zc1l_dnorm(zcl, ?T(m1p), 1)
  ?T(zc1)_c(zc1r, zc1l)
  x9_qnorm(.001, 0, 1)
  x10_qnorm(.999, 0, 1)
  x11_qnorm(.001, ?T(m1p), 1)
  x12_qnorm(.999, ?T(m1p), 1)
  ?T(x5)_seq(x9,x10, len=100)
  ?T(x6)_seq(x11,x12, len=100)
  ?T(pz0)_dnorm(?T(x5), 0, 1)
  ?T(pz1)_dnorm(?T(x6), ?T(m1p), 1)
  ?T(x56)_max(?T(x5),?T(x6))
  ?T(n56)_min(?T(x5),?T(x6))
#
# Compute Power Function
#
  ?T(px)_seq(?T(m0)-2*?T(s), ?T(m0)+2*?T(s), len=100)
  ?T(mup)_(?T(m0)-?T(px))/?T(ss)
  ?T(pwr)_1 - pnorm(zcr+?T(mup)) + pnorm(zcl+?T(mup))
  ?T(pw1)_1 - pnorm(zcr-?T(m1p)) + pnorm(zcl-?T(m1p))
#
# Compute maximum pdf for both z distribution

  ?T(zx0)_dnorm(0, 0, 1)
  ?T(zx1)_dnorm(?T(m1p), ?T(m1p), 1)
```

```
#
# Make proper label
#
   label5_encode("When MU = ",?T(m1),"   Power = ",?T(pw1))
   label_c(label, label5)
#
# Call macro to plot the screen 3.
#
   ?screen3t(?T(m1),?T(m1p),?T(x5),?T(x6),?T(pz0),?T(pz1),?T(zx0),
     ?T(zx1),?T(px),?T(zc),?T(pwr),?T(zc0),?T(zc1),
     ?T(pw1),?T(x56),?T(n56),x9,x10,x11,label)
#
# Test results
#
   ?T(pv)_ifelse(?T(x)<?T(m0), 2*pnorm(?T(x), ?T(m0), ?T(ss)),
        2*(1-pnorm(?T(x), ?T(m0), ?T(ss))))
   ?T(z)_(?T(x)-?T(m0))/?T(ss)
   label6_encode("Xbar* =",?T(x),"  Xbar crit =",xbcr,"or",xbcl)
   label7_encode("Z* =",?T(z),"  Z crit =",zcr,"or",zcl)
   label_c(label, label6, label7)
#
# Call macro to plot the screen 4.
#
   xv1_rbind(?T(x3),?T(x4))
   pxv_rbind(?T(px0),?T(px1))
   xxv_c(?T(xx0),?T(xx1))
   xcv_c(?T(xc0),?T(xc1))
   xn1_c(?T(x34),?T(n34))
   xp1_c(x5,x6,x7)
   xv2_rbind(?T(x5),?T(x6))
   pzv_rbind(?T(pz0),?T(pz1))
   zxv_c(?T(zx0),?T(zx1))
   zcv_c(?T(zc0),?T(zc1))
   xn2_c(?T(x56),?T(n56))
   xp2_c(x9,x10,x11)
   mmv_c(?T(m0),?T(m1),?T(m1p))
   ?screen4t(?T(x),?T(xbc),?T(pv),?T(zc),?T(z),
            mmv,xv1,pxv,xxv,xcv,xn1,xp1,xv2,pzv,
            zxv,zcv,xn2,xp2,label)
   rm(?T(x),?T(n),?T(s),?T(m0),?T(m1),?T(a),?T(es),?T(ss),
      ?T(x1),?T(x2),?T(x3),?T(x4),?T(x5),?T(x6),?T(x12),?T(n12),
      ?T(x34),?T(n34),?T(x56),?T(n56),?T(pm0),?T(pm1),?T(mx0),?T(mx1),
      ?T(xbc),?T(xc0),?T(xc1),?T(px0),?T(px1),?T(xx0),?T(xx1),?T(xx),
      ?T(lr),?T(k),?T(m1p),?T(pz0),?T(pz1),?T(px),?T(mup),?T(pwr),
      ?T(pw1),?T(zx0),?T(zx1),?T(pv),?T(zc),?T(zc0),?T(zc1),
      x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,label1,label2,label3,
      label4,label5,label6,label7,xv1,xv2,pxv,pzv,xxv,zxv,xcv,zcv,
      xn1,xn2,mmv,xp1,xp2,xbcr,xbcl,xc0r,xc0l,label,
      xc1r,xc1l,kr,kl,zcr,zcl,zc1r,zc1l,?T(b),?T(z))
})
END
```

```
MACRO screen1(mu0,mu1,x1,x2,pm0,pm1,mx0,mx1,mx,mn,lab1)
#
# This macro is to plot the statistical hypothesis
# on the first screen.
#
({
  par(mfrow=c(1,1), oma=c(3,2,1,0))
  plot(x1,pm0, xlab="X", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  points(x2,pm1)
  segments(mu0,mx0,mu0,0)
  segments(mu1,mx1,mu1,0)
  title(sub="Ho : smooth line    Ha : star line")
  mtext(side=2, line=1, outer=T, "PDF")
  mtext(side=3, line=0, outer=T,
   "Statistical Hypothesis")
  mtext(side=1, line=1, outer=T, lab1)
})
END
```

AD-A174 297 DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL PACKAGE (ISTP) FOR LE..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF SYST.. K H CHUL 5/5

UNCLASSIFIED SEP 86 AFIT/GSM/ENC/86S-10 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
MACRO screen2r(mu0,mu1,x3,x4,px0,px1,xx0,xx1,xx,lr,k,xbc,xc0,
            xc1,mx,mn,xb,xs,lab)
#
# This macro is to plot the likelihood ratio and
# test of the statistical hypothesis.
# ( Right tail test )
#
({
  par(mfrow=c(2,1), oma=c(4,3,0,0))
#
# Plot the Likelihood Ratio on the X bar axis
#
  plot(xx,lr, type="l", xlab="Xbar", ylab="",
       err=-1)
  segments(xbc,0,xbc,k)
  title(main="Likelihood Ratio Function",
        sub=lab[4])
#
# Plot the Hypothesis test on the X bar axis
#
  plot(x3,px0, xlab="Xbar", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  lines(x4,px1)
  mtext(side=2, line=1, outer=T,
    "    Lambda             PDF     ")
  segments(mu0,xx0,mu0,0)
  segments(mu1,xx1,mu1,0)
  abline(v=xbc)
#
# Find the alpha area
#
  al_x3[x3 > xbc]
  nal_len(al)
  al_c(xbc, al)  # X coordinates for alpha area
  ay_px0[(101-nal):100]
  ay_c(xc0, ay)  # Y coordinates for alpha area
#
# Find the beta area
#
  be_x4[x4 < xbc]
  nbe_len(be)
  be_c(be, xbc)   # X coordinates for beta area
  by_px1[1:(max(nbe,1))]
  by_c(by, xc1)   # Y coordinates for beta area
#
# Shade the beta area
#
   if(nbe>0) {
  hatch(c(xs,be,xbc), c(0,by,0), space=.05, angle=135, border=F)
   }
```

```
#
# Shade the alpha area
#
  hatch(c(xbc,al,xb), c(0,ay,0), space=.05,  border=F)
  title(main="Test of Statistical Hypothesis",
        sub="Ho : Left   Ha : Right")
  mtext(side=1, line=0, outer=T, lab[3])
  mtext(side=1, line=1, outer=T, lab[1])
  mtext(side=1, line=3, outer=T, lab[2])
  rm(al,nal,ay,be,nbe,by)
})
END
```

```
MACRO screen21(mu0,mu1,x3,x4,px0,px1,xx0,xx1,xx,lr,k,xbc,xc0,
           xc1,mx,mn,xb,xs,lab)
#
# This macro is to plot the likelihood ratio and
# test of the statistical hypothesis.
# ( Left tail test )
#
({
  par(mfrow=c(2,1), oma=c(4,3,0,0))
#
# Plot the Likelihood Ratio on the X bar axis
#
  plot(xx,lr, type="l", xlab="Xbar", ylab="",
       err=-1)
  segments(xbc,0,xbc,k)
  title(main="Likelihood Ratio Function",
        sub=lab[4])
#
# Plot the Hypothesis test on the X bar axis
#
  plot(x3,px0, xlab="Xbar", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  lines(x4,px1)
  mtext(side=2, line=1, outer=T,
    "    Lambda           PDF     ")
  segments(mu0,xx0,mu0,0)
  segments(mu1,xx1,mu1,0)
  abline(v=xbc)
#
# Find the alpha area
#
  al_x3[x3 < xbc]
  nal_len(al)
  al_c(al, xbc)  # X coordinates for alpha area
  ay_px0[1:nal]
  ay_c(ay, xc0)  # Y coordinates for alpha area
#
# Find the beta area
#
  be_x4[x4 > xbc]
  nbe_len(be)
  be_c(xbc, be)   # X coordinates for beta area
  by_px1[(min(100,(101-nbe))):100]
  by_c(xc1, by)   # Y coordinates for beta area
#
# Shade the beta area
#
  if(nbe>0) {
  hatch(c(xbc,be,xs), c(0,by,0), space=.05, angle=135, border=F)
  }
```

```
#
# Shade the alpha area
#
  hatch(c(xb,al,xbc), c(0,ay,0), space=.05,  border=F)
  title(main="Test of Statistical Hypothesis",
        sub="Ho : Right    Ha : Left")
  mtext(side=1, line=0, outer=T, lab[3])
  mtext(side=1, line=1, outer=T, lab[1])
  mtext(side=1, line=3, outer=T, lab[2])
  rm(al,nal,ay,be,nbe,by)
})
END
```

```
MACRO screen2t(mu0,mu1,x3,x4,px0,px1,xx0,xx1,xx,lr,k,xbc,
               xc0,xc1,mx,mn,xa,xb,xs,lab)
#
# This macro is to plot the likelihood ratio and
# test of the statistical hypothesis.
# ( Two tail test )
#
({
  par(mfrow=c(2,1), oma=c(4,3,0,0))
#
# Plot the Likelihood Ratio on the X bar axis
#
  plot(xx,lr, type="l", xlab="Xbar", ylab="",
       err=-1)
  segments(xbc[1],0,xbc[1],k[1])
  segments(xbc[2],0,xbc[2],k[2])
  title(main="Likelihood Ratio Function",
        sub=lab[4])
#
# Plot the Hypothesis test on the X bar axis
#
  plot(x3,px0, xlab="Xbar", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  lines(x4,px1)
  mtext(side=2, line=1, outer=T,
    "    Lambda            PDF      ")
  segments(mu0,xx0,mu0,0)
  segments(mu1,xx1,mu1,0)
  abline(v=xbc[1])
  abline(v=xbc[2])
#
# Find the alpha area
#
  apr_x3[x3 > xbc[1]]
  napr_len(apr)
  apr_c(xbc[1], apr)  # X coordinates for right side of alpha area
  apl_x3[x3 < xbc[2]]
  napl_len(apl)
  apl_c(apl, xbc[2])  # X coordinates for left side of alpha area
  ayr_px0[(101-napr):100]
  ayr_c(xc0[1], ayr)  # Y coordinates for right side of alpha area
  ayl_px0[1:napl]
  ayl_c(ayl, xc0[2])  # Y coordinates for left side of alpha area
#
# Find the beta area
#
  be_x4[(x4<xbc[1]) & (x4>xbc[2])]
  nbe_len(be)
  if(nbe>0) {
  minb_min(be)
```

```
  ibe_len(?which(x4<=minb))
  fbe_len(be)
  bxl_max(xbc[2],xs)
  bxr_max(xbc[1],xs)
  be_c(bxl, be, bxr)   # X coordinates for beta area
  by_px1[(max(1, ibe)):(max(1,( fbe+ ibe-1)))]
  miny_min(px1)
  byl_max(xc1[2],miny)
  byr_max(xc1[1],miny)
  by_c(byl, by, byr)   # Y coordinates for beta area
#
# Shade the beta area
#
  hatch(c(bxl,be,bxr), c(0,by,0), space=.05, angle=135, border=F)
  rm(minb, ibe, fbe, bxl, bxr, miny, byl, byr, by)
  }
#
# Shade the alpha area
#
  hatch(c(xa,apl,xbc[2]), c(0,ayl,0), space=.05,  border=F)
  hatch(c(xbc[1],apr,xb), c(0,ayr,0), space=.05,  border=F)
  title(main="Test of Statistical Hypothesis",
    "Ho : Left    Ha : Right")
  mtext(side=1, line=0, outer=T, lab[3])
  mtext(side=1, line=1, outer=T, lab[1])
  mtext(side=1, line=3, outer=T, lab[2])
  rm( apr,napr,apl,napl,ayr,ayl,be,nbe)
})
END
```

```
MACRO screen3r(mu1,m1p,x5,x6,pz0,pz1,zx0,zx1,px,zc,pwr,zc0,zc1,
              pw1,mx,mn,xb,xs,lab)
#
# This macro is to plot the test of the statistical hypothesis
# in Z scale and the power function.
# ( Right tail test )
#
({
  par(mfrow=c(2,1), oma=c(4,3,0,0))
#
# Plot the Power Function
#
  plot(px,pwr, type="l", xlab="MU", ylab="", err=-1)
  segments(mu1,pw1,mu1,0)
  title(main="Power Function of the Test",
        sub=lab[5])
#
# Plot the Hypothesis test on the Z axis
#
  plot(x5,pz0, xlab="Z", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  lines(x6,pz1)
  segments(0,zx0,0,0)
  segments(m1p,zx1,m1p,0)
  abline(v=zc)
#
# Find the alpha area
#
  al_x5[x5 > zc]
  nal_len(al)
  al_c(zc, al)  # X coordinates for alpha area
  ay_pz0[(101-nal):100]
  ay_c(zc0, ay)  # Y coordinates for alpha area
#
# Find the beta area
#
  be_x6[x6 < zc]
  nbe_len(be)
  be_c(be, zc)    # X coordinates for beta area
  by_pz1[1:(max(nbe,1))]
  by_c(by, zc1)    # Y coordinates for beta area
#
# Shade the beta area
#
  if(nbe>0) {
  hatch(c(xs,be,zc), c(0,by,0), space=.05, angle=135, border=F)
  }
```

```
#
# Shade the alpha area
#
  hatch(c(zc,al,xb), c(0,ay,0), space=.05, border=F)
  title(main="Test of Statistical Hypothesis",
        sub="Ho : Left    Ha : Right")
  mtext(side=2, line=1, outer=T,
    "  POWER                PDF     ")
  mtext(side=1, line=0, outer=T, lab[3])
  mtext(side=1, line=1, outer=T, lab[1])
  mtext(side=1, line=3, outer=T, lab[2])
  rm(al,nal,ay,be,nbe,by)
})
END
```

```
MACRO screen31(mu1,m1p,x5,x6,pz0,pz1,zx0,zx1,px,zc,pwr,zc0,zc1,
            pw1,mx,mn,xb,xs,lab)
#
# This macro is to plot the test of the statistical hypothesis
# in Z scale and the power function.
# ( Left tail test )
#
({
  par(mfrow=c(2,1), oma=c(4,3,0,0))
#
# Plot the Power Function
#
  plot(px,pwr, type="l", xlab="MU", ylab="", err=-1)
  segments(mu1,pw1,mu1,0)
  title(main="Power Function of the Test",
        sub=lab[5])
#
# Plot the Hypothesis test on the Z axis
#
  plot(x5,pz0, xlab="Z", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  lines(x6,pz1)
  segments(0,zx0,0,0)
  segments(m1p,zx1,m1p,0)
  abline(v=zc)
#
# Find the alpha area
#
  al_x5[x5 < zc]
  nal_len(al)
  al_c(al, zc)  # X coordinates for alpha area
  ay_pz0[1:nal]
  ay_c(ay, zc0)  # Y coordinates for alpha area
#
# Find the beta area
#
  be_x6[x6 > zc]
  nbe_len(be)
  be_c(zc, be)   # X coordinates for beta area
  by_pz1[(min(100, (101-nbe))):100]
  by_c(zc1, by)   # Y coordinates for beta area
#
# Shade the beta area
#
  if(nbe>0) {
  hatch(c(zc,be,xs), c(0,by,0), space=.05, angle=135, border=F)
  }
```

```
#
# Shade the alpha area
#
  hatch(c(xb,al,zc), c(0,ay,0), space=.05, border=F)
  title(main="Test of Statistical Hypothesis",
        sub="Ho : Right  Ha : Left")
  mtext(side=2, line=1, outer=T,
    "  POWER              PDF     ")
  mtext(side=1, line=0, outer=T, lab[3])
  mtext(side=1, line=1, outer=T, lab[1])
  mtext(side=1, line=3, outer=T, lab[2])
  rm( al,nal,ay,be,nbe,by)
})
END
```

```
MACRO screen3t(mu1,m1p,x5,x6,pz0,pz1,zx0,zx1,px,zc,pwr,zc0,zc1,
              pw1,mx,mn,za,zb,zs,lab)
#
# This macro is to plot the test of the statistical hypothesis
# in Z scale and the power function.
# ( Two tail test )
#
({
  par(mfrow=c(2,1), oma=c(4,3,0,0))
#
# Plot the Power Function
#
  plot(px,pwr, type="l", xlab="MU", ylab="", err=-1)
  segments(mu1,pw1,mu1,0)
  title(main="Power Function of the Test",
        sub=lab[5])
#
# Plot the Hypothesis test on the Z axis
#
  plot(x5,pz0, xlab="Z", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  lines(x6,pz1)
  segments(0,zx0,0,0)
  segments(m1p,zx1,m1p,0)
  abline(v=zc[1])
  abline(v=zc[2])
#
# Find the alpha area
#
  apr_x5[x5 > zc[1]]
  napr_len(apr)
  apr_c(zc[1], apr)  # X coordinates for right side of alpha area
  apl_x5[x5 < zc[2]]
  napl_len(apl)
  apl_c(apl, zc[2])  # X coordinates for left side of alpha area
  ayr_pz0[(101-napr):100]
  ayr_c(zc0[1], ayr)    # Y coordinates for right side of alpha area
  ayl_pz0[1:napl]
  ayl_c(ayl, zc0[2])    # Y coordinates for left side of alpha area
#
# Find the beta area
#
  be_x6[(x6<zc[1]) & (x6>zc[2])]
  nbe_len(be)
  if(nbe>0) {
  minb_min(be)
  ibe_len(?which(x6<=minb))
  fbe_len(be)
  bxl_max(zc[2],zs)
  bxr_max(zc[1],zs)
```

```
  be_c(bxl, be, bxr)   # X coordinates for beta area
  by_pz1[(max(1,ibe)):(max(1,(fbe+ibe-1)))]
  miny_min(pz1)
  byl_max(zc1[2],miny)
  byr_max(zc1[1],miny)
  by_c(byl, by, byr)   # Y coordinates for beta area
#
# Shade the beta area
#
  hatch(c(bxl,be,bxr), c(0,by,0), space=.05, angle=135, border=F)
  rm(minb, ibe, fbe, bxl, bxr, miny, byl, byr, by)
  }
#
# Shade the alpha area
#
  hatch(c(za,apl,zc[2]), c(0,ayl,0), space=.05,  border=F)
  hatch(c(zc[1],apr,zb), c(0,ayr,0), space=.05,  border=F)
  title(main="Test of Statistical Hypothesis",
        sub="Ho : Left  Ha : Right")
  mtext(side=2, line=1, outer=T,
    "  POWER                PDF      ")
  mtext(side=1, line=0, outer=T, lab[3])
  mtext(side=1, line=1, outer=T, lab[1])
  mtext(side=1, line=3, outer=T, lab[2])
  rm(apr,napr,apl,napl,ayr,ayl,be,nbe)
})
END
```

```
MACRO screen4r(xba,xbc,pv,zc,zs,mmv,xv1,pxv,xxv,xcv,xn1,
            bs1,xv2,pzv,zxv,zcv,xn2,bs2,lab)
#
#
# This macro is to plot the test result for right tail test.
#
({
  mu0_mmv[1]
  mu1_mmv[2]
  m1p_mmv[3]
  xx3_xv1[1,]
  xx4_xv1[2,]
  xx5_xv2[1,]
  xx6_xv2[2,]
  px0_pxv[1,]
  px1_pxv[2,]
  pz0_pzv[1,]
  pz1_pzv[2,]
  xx0_xxv[1]
  xx1_xxv[2]
  zx0_zxv[1]
  zx1_zxv[2]
  xc0_xcv[1]
  xc1_xcv[2]
  zc0_zcv[1]
  zc1_zcv[2]
  mx_xn1[1]
  mn_xn1[2]
  mxp_xn2[1]
  mnp_xn2[2]
  xb_bs1[1]
  xs_bs1[2]
  xbp_bs2[1]
  xsp_bs2[2]
  par(mfrow=c(2,1), oma=c(5,2,0,0))
#
# Plot the test result in Xbar axis.
#
  plot(xx3,px0, xlab="Xbar", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  lines(xx4,px1)
  mtext(side=2, line=1, outer=T,
  "   PDF              PDF   ")
  segments(mu0,xx0,mu0,0)
  segments(mu1,xx1,mu1,0)
  abline(v=xbc)
#
# Find the alpha area
#
  al_xx3[xx3 > xbc]
  nal_len(al)
  al_c(xbc, al)  # X coordinates for alpha area
```

```
  ay_px0[(101-nal):100]
  ay_c(xc0, ay)  # Y coordinates for alpha area
#
# Find the beta area
#
  be_xx4[xx4 < xbc]
  nbe_len(be)
  be_c(be, xbc)   # X coordinates for beta area
  by_px1[1:(max(nbe,1))]
  by_c(by, xc1)   # Y coordinates for beta area
#
# Shade the beta area
#
  if(nbe>0) {
  hatch(c(xs,be,xbc), c(0,by,0), space=.05, angle=135, border=F)
  }
#
# Shade the alpha area
#
  hatch(c(xbc,al,xb), c(0,ay,0), space=.05, border=F)
#
# Indicate the Xbar*
#
  arrows(xba,xx0/2,xba,0)
  title(main="Result of Statistical Test", sub=lab[3])
#
# Plot the test result in Z axis.
#
  plot(xx5,pz0, xlab="Z", ylab="", type="l",
       xlim=c(mnp,mxp),err=-1)
  lines(xx6,pz1)
  segments(0,zx0,0,0)
  segments(m1p,zx1,m1p,0)
  abline(v=zc)
#
# Find the alpha area
#
  al_xx5[xx5 > zc]
  nal_len(al)
  al_c(zc, al)  # X coordinates for alpha area
  ay_pz0[(101-nal):100]
  ay_c(zc0, ay)  # Y coordinates for alpha area
#
# Find the beta area
#
  be_xx6[xx6 < zc]
  nbe_len(be)
  be_c(be, zc)   # X coordinates for beta area
  by_pz1[1:(max(nbe,1))]
  by_c(by, zc1)   # Y coordinates for beta area
```

```
#
# Shade the beta area
#
  if(nbe>0) {
  hatch(c(xsp,be,zc), c(0,by,0), space=.05, angle=135, border=F)
  }
#
# Shade the alpha area
#
  hatch(c(zc,al,xbp), c(0,ay,0), space=.05, border=F)
#
# Indicate the Z*
#
  arrows(zs,zx0/2,zs,0)
  title(sub="Ho : Left    Ha : Right")
  mtext(side=1, line=0, outer=T, lab[1])
  mtext(side=1, line=2, outer=T, lab[2])
  mtext(side=1, line=3, outer=T, lab[6])
  dcs_ifelse(xba>xbc, "Reject Ho", "Do Not Reject Ho")
  mtext(side=1, line=4, outer=T,
     encode("Decision : ",dcs,"   P-value =",pv))
  rm(al,nal,ay,be,nbe,by,dcs,mu0,mu1,m1p,xx3,xx4,xx5,xx6,px0,
     px1,pz0,pz1,xx0,xx1,zx0,zx1,xc0,xc1,zc0,zc1,mx,mn,
     mxp,mnp,xb,xs,xbp,xsp)
})
END
```

```
MACRO screen41(xba,xbc,pv,zc,zs,mmv,xv1,pxv,xxv,xcv,xn1,
           bs1,xv2,pzv,zxv,zcv,xn2,bs2,lab)
#
#
# This macro is to plot the test result for left tail test.
#
({
  mu0_mmv[1]
  mu1_mmv[2]
  m1p_mmv[3]
  xx3_xv1[1,]
  xx4_xv1[2,]
  xx5_xv2[1,]
  xx6_xv2[2,]
  px0_pxv[1,]
  px1_pxv[2,]
  pz0_pzv[1,]
  pz1_pzv[2,]
  xx0_xxv[1]
  xx1_xxv[2]
  zx0_zxv[1]
  zx1_zxv[2]
  xc0_xcv[1]
  xc1_xcv[2]
  zc0_zcv[1]
  zc1_zcv[2]
  mx_xn1[1]
  mn_xn1[2]
  mxp_xn2[1]
  mnp_xn2[2]
  xb_bs1[1]
  xs_bs1[2]
  xbp_bs2[1]
  xsp_bs2[2]
  par(mfrow=c(2,1), oma=c(5,2,0,0))
#
# Test result in the Xbar axis.
#
  plot(xx3,px0, xlab="Xbar", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  lines(xx4,px1)
  mtext(side=2, line=1, outer=T,
  "   PDF              PDF   ")
  segments(mu0,xx0,mu0,0)
  segments(mu1,xx1,mu1,0)
  abline(v=xbc)
```

```
#
# Find the alpha area
#
  al_xx3[xx3 < xbc]
  nal_len(al)
  al_c(al, xbc)  # X coordinates for alpha area
  ay_px0[1:nal]
  ay_c(ay, xc0)  # Y coordinates for alpha area
#
# Find the beta area
#
  be_xx4[xx4 > xbc]
  nbe_len(be)
  be_c(xbc, be)   # X coordinates for beta area
  by_px1[(min(100, (101-nbe))) : 100]
  by_c(xc1, by)   # Y coordinates for beta area
#
# Shade the beta area
#
  if(nbe>0) {
  hatch(c(xbc, be, xs), c(0,by,0), space=.05, angle=135, border=F)
  }
#
# Shade the alpha area
#
  hatch(c(xb,al,xbc), c(0,ay,0),  space=.05, border=F)
#
# Indicate the Xbar*
#
  arrows(xba,xx0/2,xba,0)
  title(main="Result of Statistical Test", sub=lab[3])
#
# Test result in Z axis.
#
  plot(xx5,pz0, xlab="Z", ylab="", type="l",
       xlim=c(mnp,mxp),err=-1)
  lines(xx6,pz1)
  segments(0,zx0,0,0)
  segments(m1p,zx1,m1p,0)
  abline(v=zc)
#
# Find the alpha area
#
  al_xx5[xx5 < zc]
  nal_len(al)
  al_c(al, zc)  # X coordinates for alpha area
  ay_pz0[1:nal]
  ay_c(ay, zc0)  # Y coordinates for alpha area
```

```
#
# Find the beta area
#
   be_xx6[xx6 > zc]
   nbe_len(be)
   be_c(zc, be)    # X coordinates for beta area
   by_pz1[(min(100, (101-nbe))):100]
   by_c(zc1, by)    # Y coordinates for beta area
#
# Shade the beta area
#
   if(nbe>0) {
   hatch(c(zc,be,xsp), c(0,by,0), space=.05, angle=135, border=F)
   }
#
# Shade the alpha area
#
   hatch(c(xbp,al,zc), c(0,ay,0), space=.05, border=F)
#
# Indicate the Z*
#
   arrows(zs,zx0/2,zs,0)
   title(sub="Ho : Right    Ha : Left")
   mtext(side=1, line=0, outer=T, lab[1])
   mtext(side=1, line=2, outer=T, lab[2])
   mtext(side=1, line=3, outer=T, lab[6])
   dcs_ifelse(xba>xbc, "Do Not Reject Ho", "Reject Ho")
   mtext(side=1, line=4, outer=T,
      encode("Decision : ",dcs,"   P-value =",pv))
   rm(al,nal,ay,be,nbe,by,dcs,mu0,mu1,m1p,xx3,xx4,xx5,xx6,
      px0,px1,pz0,pz1,xx0,xx1,zx0,zx1,xc0,xc1,zc0,zc1,
      mx,mn,mxp,mnp,xb,xs,xbp,xsp)
})
END
```

```
MACRO screen4t(xba,xbc,pv,zc,zs,mmv,xv1,pxv,xxv,xcv,xn1,
          bs1,xv2,pzv,zxv,zcv,xn2,bs2,lab)
#
# This macro is to plot the test result for two tail test.
#
({
  mu0_mmv[1]
  mu1_mmv[2]
  m1p_mmv[3]
  xx3_xv1[1,]
  xx4_xv1[2,]
  xx5_xv2[1,]
  xx6_xv2[2,]
  px0_pxv[1,]
  px1_pxv[2,]
  pz0_pzv[1,]
  pz1_pzv[2,]
  xx0_xxv[1]
  xx1_xxv[2]
  zx0_zxv[1]
  zx1_zxv[2]
  xc0_xcv[1:2]
  xc1_xcv[3:4]
  zc0_zcv[1:2]
  zc1_zcv[3:4]
  mx_xn1[1]
  mn_xn1[2]
  mxp_xn2[1]
  mnp_xn2[2]
  xa_bs1[1]
  xb_bs1[2]
  xs_bs1[3]
  za_bs2[1]
  zb_bs2[2]
  zd_bs2[3]
  par(mfrow=c(2,1), oma=c(5,2,0,0))
#
# Test result in Xbar axis.
#
  plot(xx3,px0, xlab="Xbar", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  lines(xx4,px1)
  mtext(side=2, line=1, outer=T,
  "   PDF              PDF    ")
  segments(mu0,xx0,mu0,0)
  segments(mu1,xx1,mu1,0)
  abline(v=xbc)
```

```
#
# Find the alpha area
#
  apr_xx3[xx3 > xbc[1]]
  napr_len(apr)
  apr_c(xbc[1], apr)  # X coordinates for right side of alpha area
  apl_xx3[xx3 < xbc[2]]
  napl_len(apl)
  apl_c(apl, xbc[2])  # X coordinates for left side of alpha area
  ayr_px0[(101-napr):100]
  ayr_c(xc0[1], ayr)  # Y coordinates for right side of alpha area
  ayl_px0[1:napl]
  ayl_c(ayl, xc0[2])  # Y coordinates for left side of alpha area
#
# Find the beta area
#
  be_xx4[(xx4<xbc[1]) & (xx4>xbc[2])]
  nbe_len(be)
  if(nbe>0) {
  minb_min(be)
  ibe_len(?which(xx4<=minb))
  fbe_len(be)
  bxl_max(xbc[2],xs)
  bxr_max(xbc[1],xs)
  be_c(bxl, be, bxr)   # X coordinates for beta area
  by_px1[(max(1,ibe)):(max(1,(fbe+ibe-1)))]
  miny_min(px1)
  byl_max(xc1[2],miny)
  byr_max(xc1[1],miny)
  by_c(byl, by, byr)   # Y coordinates for beta area
#
# Shade the beta area
#
  hatch(c(bxl,be,bxr), c(0,by,0), space=.05, angle=135, border=F)
  rm(minb, ibe, fbe, bxl, bxr, miny, byl, byr, by)
  }
#
# Shade the alpha area
#
  hatch(c(xa,apl,xbc[2]), c(0,ayl,0),  space=.05, border=F)
  hatch(c(xbc[1],apr,xb), c(0,ayr,0),  space=.05, border=F)
#
# Indicate the Xbar*
#
  arrows(xba,xx0/2,xba,0)
  title(main="Result of Statistical Test", sub=lab[1])
```

```
#
# Test result in Z axis.
#
  plot(xx5,pz0, xlab="Z", ylab="", type="l",
       xlim=c(mnp,mxp),err=-1)
  lines(xx6,pz1)
  segments(0,zx0,0,0)
  segments(m1p,zx1,m1p,0)
  abline(v=zc[1])
  abline(v=zc[2])
#
# Find the alpha area
#
  apr_xx5[xx5 > zc[1]]
  napr_len(apr)
  apr_c(zc[1], apr)  # X coordinates for right side of alpha area
  apl_xx5[xx5 < zc[2]]
  napl_len(apl)
  apl_c(apl, zc[2])  # X coordinates for left side of alpha area
  ayr_pz0[(101-napr):100]
  ayr_c(zc0[1], ayr)    # Y coordinates for right side of alpha area
  ayl_pz0[1:napl]
  ayl_c(ayl, zc0[2])    # Y coordinates for left side of alpha area
#
# Find the beta area
#
  be_xx6[(xx6<zc[1]) & (xx6>zc[2])]
  nbe_len(be)
  if(nbe>0) {
  minb_min(be)
  ibe_len(?which(xx6<=minb))
  fbe_len(be)
  bxl_max(zc[2],zd)
  bxr_max(zc[1],zd)
  be_c(bxl, be, bxr)   # X coordinates for beta area
  by_pz1[(max(1, ibe)):(max(1,(fbe+ibe-1)))]
  miny_min(pz1)
  byl_max(zc1[2],miny)
  byr_max(zc1[1],miny)
  by_c(byl, by, byr)   # Y coordinates for beta area
#
# Shade the beta area
#
  hatch(c(bxl,be,bxr), c(0,by,0), space=.05, angle=135, border=F)
  rm(minb, ibe, fbe, bxl, bxr, miny, byl, byr, by)
  }
```

```
#
# Shade the alpha area
#
  hatch(c(za,apl,zc[2]), c(0,ayl,0), space=.05,  border=F)
  hatch(c(zc[1],apr,zb), c(0,ayr,0), space=.05,  border=F)
#
# Indicate the Z*
#
  arrows(zs,zx0/2,zs,0)
  title(main=lab[3],sub="Ho : Left    Ha : Right")
  mtext(side=1, line=0, outer=T, lab[1])
  mtext(side=1, line=1, outer=T, lab[2])
  mtext(side=1, line=2, outer=T, lab[6])
  mtext(side=1, line=3, outer=T, lab[7])
  dcs_ifelse(((xba>xbc[1]) | (xba<xbc[2])), "Reject Ho",
     "Do Not Reject Ho")
  mtext(side=1, line=4, outer=T,
     encode("Decision : ",dcs,"   P-value =",pv))
  rm( apr,napr,apl,napl,ayr,ayl,be,nbe,dcs,mu0,mu1,
     m1p,xx3,xx4,xx5,xx6,px0,px1,pz0,pz1,xx0,xx1,zx0,
     zx1,xc0,xc1,zc0,zc1,mx,mn,mxp,mnp,xa,xb,xs,za,zb,
     zd)
})
END
```

```
MACRO test2
({
  ?MESSAGE()
  ?MESSAGE(Null Hypothesis  Ho :  MU = MUo)
  ?MESSAGE(Please select proper alternative hypothesis.)
  ?MESSAGE()
  item_c(" Ha :  MU  >  MUo  (Right tail test)",
         " Ha :  MU  <  MUo  (Left  tail test)",
         " Ha :  MU !=  MUo  (Two   tail test)")
  action_c("?test2r",
           "?test2l",
           "?test2t")
  menu( item, action)
  rm( item, action)
})
END
```

```
MACRO test2r(
xb/?PROMPT(X bar       : )/,
nv/?PROMPT(Sample Size : )/,
sv/?PROMPT(Sample S.D. : )/,
m0/?PROMPT(MUo         : )/,
al/?PROMPT(Alpha       : )/,
es/?PROMPT(Effect Size for computing power : )/)
chapter
({
#
# This macro is to test for a mean for a normal population
# with unknown sigma.
# ( Right tail test )
#
   ?MESSAGE()
   ?T(xb)_xb
   ?T(n)_nv
   ?T(s)_sv
   ?T(m0)_m0
   ?T(a)_al
   ?MESSAGE()
   ?MESSAGE(Effect size is the distance between MUo and Mu1,)
   ?MESSAGE(and it is absolute value.)
   ?MESSAGE()
   ?T(es)_es
   ?T(ss)_?T(s)/sqrt(?T(n))
   ?T(m1)_?T(m0)+?T(es)
#
# Compute quantile for following populations from cdf=.001 to .999;
#    normal population : N(MUo, sigma)  & N(MUo, sigma/sqrt(n))
#                        N(MU1, sigma)  & N(MU1, sigma/sqrt(n))
#
   x1_qnorm(.001, ?T(m0), ?T(s))
   x2_qnorm(.999, ?T(m0), ?T(s))
   x3_qnorm(.001, ?T(m1), ?T(s))
   x4_qnorm(.999, ?T(m1), ?T(s))
   ?T(x1)_seq(x1,x2, len=100)
   ?T(x2)_seq(x3,x4, len=100)
   ?T(x12)_max(?T(x1),?T(x2))
   ?T(n12)_min(?T(x1),?T(x2))
#
# Compute PDF for normal population.
#
   ?T(pm0)_dnorm(?T(x1), ?T(m0), ?T(s))
   ?T(pm1)_dnorm(?T(x2), ?T(m1), ?T(s))
#
# Compute maximum pdf for both normal distribution
#
   ?T(mx0)_dnorm(?T(m0), ?T(m0), ?T(s))
   ?T(mx1)_dnorm(?T(m1), ?T(m1), ?T(s))
```

```
#
# Make proper label
#
   label1_encode("  MUo =",?T(m0),"  MU1 =", ?T(m1),
         " Estimated sigma =",?T(s),"  Effect Size = ",?T(es))
#
# Call macro to plot the screen 1.
#
   ?screen1p(?T(m0),?T(m1),?T(x1),?T(x2),?T(pm0),?T(pm1),
          ?T(mx0),?T(mx1),?T(x12),?T(n12),label1)
#
# Compute PDF in t scale
#
   ?T(ncp)_(?T(m1)-?T(m0))/?T(ss)     # noncentral t parameter
   ?T(df)_?T(n)-1                     # Degree of freedom
   ?T(tc)_qt((1-?T(a)),?T(df))        # t critical value
   ?T(tc0)_dt(?T(tc), ?T(df))         # PDF at t critical value under
                                      #   the null distribution
   x5_qt(.001, ?T(df))
   x6_qt(.999, ?T(df))
   ?T(x3)_seq(x5,x6, len=100)         # T quantile for null distribution
   ?T(pt0)_dt(?T(x3), ?T(df))         # PDF for null distribution
   nctout_nct(?T(ncp),?T(df),6)
   ?T(x4)_nctout$ord                  # T quantile coresponding to PDF
   ?T(pt1)_nctout$density             # PDF for alternative distribution
   ?T(x34)_max(?T(x3),?T(x4))         # maximum value on the x-axis
   ?T(n34)_min(?T(x3),?T(x4))         # minimum value on the x-axis
   ?T(xn)_c(?T(x34), ?T(n34))
#
# Compute likelihood ratio on the t-axis
#
   ?T(lr)_(1+((?T(x3)^2)*(1/?T(df))))^(-?T(n)/2)
   ?T(k)_(1+((?T(tc)^2)*(1/?T(df))))^(-?T(n)/2)  # Likelihood ratio
                                     # at the t-critical value
#
# Find the pdf which is the closest to the pdf of t-critical value
# to shade th beta area by finding the position in the vector.
#
   nnct_len(?which(?T(x4)<=?T(tc)))
   ?T(tc1)_?T(pt1)[max(1,nnct)]
#
# Compute Power Function
#
   pwrmu_seq(?T(m0)-?T(s), ?T(m0)+2*?T(s), len=60)
   pwrncp_(pwrmu-?T(m0))/?T(ss)
   pwrcdf_nctp(pwrncp,?T(df),?T(tc))
   ?T(pwr)_1-pwrcdf
#
# Compute beta when mu = mu1
#
   ?T(b)_nctp(?T(ncp),?T(df),?T(tc))
   ?T(pw1)_1-?T(b)
```

```
#
# Compute maximum pdf for both t distribution
#
   ?T(tx0)_dt(0,?T(df))    # max pdf under the null distribution
   ?T(tx1)_max(?T(pt1))    # max pdf under alternative distribution
#
# Test results
#
   ?T(ts)_(?T(xb)-?T(m0))/?T(ss)
   ?T(pv)_1-pt(?T(ts), ?T(df))
#
# Make proper label
#
   label1_encode("  MUo =",?T(m0),"  MU1 =", ?T(m1),
           " Estimated sigma =",?T(s))
   label2_encode("Effect Size = ",?T(es)," Sample size =", ?T(n))
   label3_encode("//// : Alpha  (",?T(a),")   \\\\\\\\\\\\\\\\",
                 "Beta  (",?T(b),")")
   label4_encode("T-critical :",?T(tc),"  LR :",?T(k))
   label5_encode("T * = ",?T(ts),"     T crit = ",?T(tc))
   label6_encode("When MU = ",?T(m1),"   Power = ",?T(pw1))
   label_c(label1,label2,label3,label4,label5,label6)
#
# Call macro to plot screen2 and screen3
#
   ?screen23rp(?T(ncp),?T(tc),?T(tc0),?T(tc1),?T(x3),?T(x4),
              ?T(pt0),?T(pt1),?T(xn),?T(lr),?T(k),?T(tx0),?T(tx1),
              pwrmu,?T(pwr),?T(pw1),?T(m1),?T(ts),?T(pv),label)
#
   rm(?T(m0),?T(m1),?T(a),?T(es),?T(xb),?T(s),?T(n),?T(ss),
      x1,x2,x3,x4,x5,x6,?T(x1),?T(x2),?T(x3),?T(x4),?T(x12),
      ?T(n12),label1,label2,label3,label4,label5,label6,label,
      ?T(pt0),?T(pt1),?T(lr),?T(k),?T(ncp),?T(df),?T(tc),
      ?T(tc0),?T(tc1),nctout,?T(x34),?T(n34),nnct,pwrmu,?T(xn),
      pwrncp,pwrcdf,?T(pwr),?T(b),?T(pw1),?T(tx0),?T(tx1),
      ?T(ts),?T(pm0),?T(pm1),?T(mx0),?T(pv),?T(mx1))
})
END
```

```
MACRO test21(
xb/?PROMPT(X bar        : )/,
nv/?PROMPT(Sample Size : )/,
sv/?PROMPT(Sample S.D. : )/,
m0/?PROMPT(MUo          : )/,
al/?PROMPT(Alpha        : )/,
es/?PROMPT(Effect Size for computing power : )/)
chapter
({
#
# This macro is to test for a mean for a normal population
# with unknown sigma.
# ( Left tail test )
#
   ?MESSAGE()
   ?T(xb)_xb
   ?T(n)_nv
   ?T(s)_sv
   ?T(m0)_m0
   ?T(a)_al
   ?MESSAGE()
   ?MESSAGE(Effect size is the distance between MUo and Mu1,)
   ?MESSAGE(and it is absolute value.)
   ?MESSAGE()
   ?T(es)_es
   ?T(ss)_?T(s)/sqrt(?T(n))
   ?T(m1)_?T(m0)-?T(es)
#
# Compute quantile for following populations from cdf=.001 to .999;
#    normal population : N(MUo, sigma)  & N(MUo, sigma/sqrt(n))
#                        N(MU1, sigma)  & N(MU1, sigma/sqrt(n))
#
   x1_qnorm(.001, ?T(m0), ?T(s))
   x2_qnorm(.999, ?T(m0), ?T(s))
   x3_qnorm(.001, ?T(m1), ?T(s))
   x4_qnorm(.999, ?T(m1), ?T(s))
   ?T(x1)_seq(x1,x2, len=100)
   ?T(x2)_seq(x3,x4, len=100)
   ?T(x12)_max(?T(x1),?T(x2))
   ?T(n12)_min(?T(x1),?T(x2))
#
# Compute PDF for normal population.
#
   ?T(pm0)_dnorm(?T(x1), ?T(m0), ?T(s))
   ?T(pm1)_dnorm(?T(x2), ?T(m1), ?T(s))
#
# Compute maximum pdf for both normal distribution
#
   ?T(mx0)_dnorm(?T(m0), ?T(m0), ?T(s))
   ?T(mx1)_dnorm(?T(m1), ?T(m1), ?T(s))
```

```
#
# Make proper label
#
   label1_encode("  MUo =",?T(m0),"  MU1 =", ?T(m1),
         " Estimated sigma =",?T(s),"  Effect Size = ",?T(es))
#
# Call macro to plot the screen 1.
#
   ?screen1p(?T(m0),?T(m1),?T(x1),?T(x2),?T(pm0),?T(pm1),
          ?T(mx0),?T(mx1),?T(x12),?T(n12),label1)
#
# Compute PDF in t scale
#
   ?T(ncp)_(?T(m1)-?T(m0))/?T(ss)    # noncentral t parameter
   ?T(df)_?T(n)-1                    # Degree of freedom
   ?T(tc)_qt(?T(a),?T(df))           # t critical value
   ?T(tc0)_dt(?T(tc), ?T(df))        # PDF at t critical value under
                                     #   the null distribution
   x5_qt(.001, ?T(df))
   x6_qt(.999, ?T(df))
   ?T(x3)_seq(x5,x6, len=100)        # T quantile for null distribution
   ?T(pt0)_dt(?T(x3), ?T(df))        # PDF for null distribution
   nctout_nct(?T(ncp),?T(df),6)
   ?T(x4)_nctout$ord                 # T quantile coresponding to PDF
   ?T(pt1)_nctout$density            # PDF for alternative distribution
   ?T(x34)_max(?T(x3),?T(x4))        # maximum value on the x-axis
   ?T(n34)_min(?T(x3),?T(x4))        # minimum value on the x-axis
   ?T(xn)_c(?T(x34), ?T(n34))
#
# Compute likelihood ratio on the t-axis
#
   ?T(lr)_(1+((?T(x3)^2)*(1/?T(df))))^(-?T(n)/2)
   ?T(k)_(1+((?T(tc)^2)*(1/?T(df))))^(-?T(n)/2)  # Likelihood ratio
                                    # at the t-critical value
#
# Find the pdf which is the closest to the pdf of t-critical value
# to shade th beta area by finding the position in the vector.
#
   nnct_len(?which(?T(x4)<=?T(tc)))
   ?T(tc1)_?T(pt1)[max(1,nnct)]
#
# Compute Power Function
#
   pwrmu_seq(?T(m0)-2*?T(s), ?T(m0)+?T(s), len=60)
   pwrncp_(pwrmu-?T(m0))/?T(ss)
   ?T(pwr)_nctp(pwrncp,?T(df),?T(tc))
#
# Compute beta when mu = mu1
#
   ?T(pw1)_nctp(?T(ncp),?T(df),?T(tc))
   ?T(b)_1-?T(pw1)
```

```
#
# Compute maximum pdf for both t distribution
#
   ?T(tx0)_dt(0,?T(df))    # max pdf under the null distribution
   ?T(tx1)_max(?T(pt1))    # max pdf under alternative distribution
#
# Test results
#
   ?T(ts)_(?T(xb)-?T(m0))/?T(ss)
   ?T(pv)_pt(?T(ts), ?T(df))
#
# Make proper label
#
   label1_encode("  MUo =",?T(m0),"  MU1 =", ?T(m1),
          " Estimated sigma =",?T(s))
   label2_encode("Effect Size = ",?T(es)," Sample size =", ?T(n))
   label3_encode("//// : Alpha  (",?T(a),")   \\\\\\\\\\\\\\\\",
                "Beta  (",?T(b),")")
   label4_encode("T-critical :",?T(tc),"  LR :",?T(k))
   label5_encode("T # = ",?T(ts),"    T crit = ",?T(tc))
   label6_encode("When MU = ",?T(m1),"   Power = ",?T(pw1))
   label_c(label1,label2,label3,label4,label5,label6)
#
# Call macro to plot screen2 and screen3
#
   ?screen231p(?T(ncp),?T(tc),?T(tc0),?T(tc1),?T(x3),?T(x4),
               ?T(pt0),?T(pt1),?T(xn),?T(lr),?T(k),?T(tx0),?T(tx1),
               pwrmu,?T(pwr),?T(pw1),?T(m1),?T(ts),?T(pv),label)
#
   rm(?T(m0),?T(m1),?T(a),?T(es),?T(xb),?T(s),?T(n),?T(ss),
      x1,x2,x3,x4,x5,x6,?T(x1),?T(x2),?T(x3),?T(x4),?T(x12),
      ?T(n12),label1,label2,label3,label4,label5,label6,label,
      ?T(pt0),?T(pt1),?T(lr),?T(k),?T(ncp),?T(df),?T(tc),
      ?T(tc0),?T(tc1),nctout,?T(x34),?T(n34),nnct,pwrmu,
      pwrncp,?T(pwr),?T(b),?T(pw1),?T(tx0),?T(tx1),?T(xn),
      ?T(ts),?T(pm0),?T(pm1),?T(mx0),?T(pv),?T(mx1))
})
END
```

```
MACRO test2t(
xb/?PROMPT(X bar       : )/,
nv/?PROMPT(Sample Size : )/,
sv/?PROMPT(Sample S.D. : )/,
m0/?PROMPT(MUo         : )/,
al/?PROMPT(Alpha       : )/,
es/?PROMPT(Effect Size for computing power : )/)
chapter
({
#
# This macro is to test for a mean for a normal population
# with unknown sigma.
# ( Two tail test )
#
   ?MESSAGE()
   ?MESSAGE(For two tail test, alternative distribution should)
   ?MESSAGE(exist in both sides of the null distribution, however,)
   ?MESSAGE(for simplification of plotting, only right side )
   ?MESSAGE(alternative distribution will be plotted.)
   ?MESSAGE()
   ?T(xb)_xb
   ?T(n)_nv
   ?T(s)_sv
   ?T(m0)_m0
   ?T(a)_al
   ?MESSAGE()
   ?MESSAGE(Effect size is the distance between MUo and Mu1,)
   ?MESSAGE(and it is absolute value.)
   ?MESSAGE()
   ?T(es)_abs(es)
   ?T(ss)_?T(s)/sqrt(?T(n))
   ?T(m1)_?T(m0)+?T(es)
#
# Compute quantile for following populations from cdf=.001 to .999;
#    normal population : N(MUo, sigma)  & N(MUo, sigma/sqrt(n))
#                        N(MU1, sigma)  & N(MU1, sigma/sqrt(n))
#
   x1_qnorm(.001, ?T(m0), ?T(s))
   x2_qnorm(.999, ?T(m0), ?T(s))
   x3_qnorm(.001, ?T(m1), ?T(s))
   x4_qnorm(.999, ?T(m1), ?T(s))
   ?T(x1)_seq(x1,x2, len=100)
   ?T(x2)_seq(x3,x4, len=100)
   ?T(x12)_max(?T(x1),?T(x2))
   ?T(n12)_min(?T(x1),?T(x2))
#
# Compute PDF for normal population.
#
   ?T(pm0)_dnorm(?T(x1), ?T(m0), ?T(s))
   ?T(pm1)_dnorm(?T(x2), ?T(m1), ?T(s))
```

```
#
# Compute maximum pdf for both normal distribution
#
   ?T(mx0)_dnorm(?T(m0), ?T(m0), ?T(s))
   ?T(mx1)_dnorm(?T(m1), ?T(m1), ?T(s))
#
# Make proper label
#
   label1_encode("  MUo =",?T(m0),"  MU1 =", ?T(m1),
         " Estimated sigma =",?T(s),"  Effect Size = ",?T(es))
#
# Call macro to plot the screen 1.
#
   ?screen1p(?T(m0),?T(m1),?T(x1),?T(x2),?T(pm0),?T(pm1),
          ?T(mx0),?T(mx1),?T(x12),?T(n12),label1)
#
# Compute PDF in t scale
#
   ?T(ncp)_(?T(m1)-?T(m0))/?T(ss)    # noncentral t parameter
   ?T(df)_?T(n)-1                    # Degree of freedom
   twoal_c((1-(?T(a)/2)), ?T(a)/2)
   ?T(tc)_qt(twoal,?T(df))           # t critical value
# ... Two tail test has two t-critical values, and here the
# first one is right side of the t-critical value, and the second
# one is the other.
#
   ?T(tc0)_dt(?T(tc), ?T(df))        # PDF at t critical value under
                                     #   the null distribution
   x5_qt(.001, ?T(df))
   x6_qt(.999, ?T(df))
   ?T(x3)_seq(x5,x6, len=100)        # T quantile for null distribution
   ?T(pt0)_dt(?T(x3), ?T(df))        # PDF for null distribution
   nctout_nct(?T(ncp),?T(df),6)
   ?T(x4)_nctout$ord                 # T quantile coresponding to PDF
   ?T(pt1)_nctout$density            # PDF for alternative distribution
   ?T(x34)_max(?T(x3),?T(x4))        # maximum value on the x-axis
   ?T(n34)_min(?T(x3),?T(x4))        # minimum value on the x-axis
   ?T(xn)_c(?T(x34), ?T(n34))
#
# Compute likelihood ratio on the t-axis
#
   ?T(lr)_(1+((?T(x3)^2)*(1/?T(df))))^(-?T(n)/2)
   ?T(k)_(1+((?T(tc)^2)*(1/?T(df))))^(-?T(n)/2)  # Likelihood ratio
                                     # at the t-critical value
#
# Find the pdf which is the closest to the pdf of t-critical value
# to shade th beta area by finding the position in the vector.
#
   nnctr_len(?which(?T(x4)<=?T(tc)[1]))
   nnctl_len(?which(?T(x4)<=?T(tc)[2]))
```

```
    tc1r_?T(pt1)[max(1,nnctr)]
    tc1l_?T(pt1)[max(1,nnctl)]
    ?T(tc1)_c(tc1r, tc1l)
#
# Compute Power Function
#
    pwrmu_seq(?T(m0)-?T(s), ?T(m0)+2*?T(s), len=60)
    pwrncp_(pwrmu-?T(m0))/?T(ss)
    pwrcdfr_nctp(pwrncp,?T(df),?T(tc)[1])
    pwrcdfl_nctp(pwrncp,?T(df),?T(tc)[2])
    ?T(pwr)_1-pwrcdfr+pwrcdfl
#
# Compute beta when mu = mu1
#
    betar_nctp(?T(ncp),?T(df),?T(tc)[1])
    betal_nctp(?T(ncp),?T(df),?T(tc)[2])
    ?T(b)_betar-betal
    ?T(pw1)_1-?T(b)
#
# Compute maximum pdf for both t distribution
#
    ?T(tx0)_dt(0,?T(df))    # max pdf under the null distribution
    ?T(tx1)_max(?T(pt1))    # max pdf under alternative distribution
#
# Test results
#
    ?T(ts)_(?T(xb)-?T(m0))/?T(ss)
    ?T(pv)_ifelse(?T(xb)<?T(m0),2*(pt(?T(ts), ?T(df))),
          2*(1-(pt(?T(ts), ?T(df)))))
#
# Make proper label
#
    label1_encode("  MUo =",?T(m0),"  MU1 =", ?T(m1),
            " Estimated sigma =",?T(s))
    label2_encode("Effect Size = ",?T(es)," Sample size =", ?T(n))
    label3_encode("//// : Alpha  (",?T(a),")   \\\\\\\\\\\\\\\\",
              "Beta  (",?T(b),")")
    label4_encode("T-critical :",?T(tc)[1],"or",?T(tc)[2],"  LR
:",?T(k)[1])
    label5_encode("T * = ",?T(ts),"   T crit =
",?T(tc)[1],"or",?T(tc)[2])
    label6_encode("When MU = ",?T(m1),"   Power = ",?T(pw1))
    label_c(label1,label2,label3,label4,label5,label6)
#
# Call macro to plot screen2 and screen3
#
    ?screen23tp(?T(ncp),?T(tc),?T(tc0),?T(tc1),?T(x3),?T(x4),
            ?T(pt0),?T(pt1),?T(xn),?T(lr),?T(k),
            ?T(tx0),?T(tx1),pwrmu,?T(pwr),?T(pw1),?T(m1),
            ?T(ts),?T(pv),label)
```

```
   rm(?T(m0),?T(m1),?T(a),?T(es),?T(xb),?T(s),?T(n),?T(ss),
      x1,x2,x3,x4,x5,x6,?T(x1),?T(x2),?T(x3),?T(x4),?T(x12),
      ?T(n12),label1,label2,label3,label4,label5,label6,label,
      ?T(pt0),?T(pt1),?T(lr),?T(k),?T(ncp),?T(df),?T(tc),
      ?T(tc0),?T(tc1),nctout,?T(x34),?T(n34),nnctr,nnctl,
      tc1r,tc1l,pwrmu,pwrcdfr,pwrcdfl,betar,betal,twoal,
      pwrncp,?T(pwr),?T(b),?T(pw1),?T(tx0),?T(tx1),
      ?T(ts),?T(pm0),?T(pm1),?T(mx0),?T(pv),?T(mx1),?T(xn))
})
END
```

```
MACRO screen1p(mu0,mu1,x1,x2,pm0,pm1,mx0,mx1,mx,mn,lab1)
#
# This macro is to plot the statistical hypothesis for
# unknown sigma on the first screen.
#
({
  par(mfrow=c(1,1), oma=c(5,2,1,0))
  plot(x1,pm0, xlab="X", ylab="", type="l",
       xlim=c(mn,mx),err=-1)
  points(x2,pm1)
  segments(mu0,mx0,mu0,0)
  segments(mu1,mx1,mu1,0)
  title(sub="Ho : smooth line    Ha : star line")
  mtext(side=2, line=1, outer=T, "PDF")
  mtext(side=3, line=0, outer=T,
   "Statistical Hypothesis")
  mtext(side=1, line=1, outer=T, lab1)
  mtext(side=1, line=3, outer=T,
   "Note : Alternative hypothesis is estimated as normal
   with unknown parameters")
})
END
```

```
MACRO screen23rp(ncp,tc,tc0,tc1,x3,x4,pt0,pt1,mxn,lr,kv,
               tx0,tx1,pwrmu,pwr,pw1,mu1,ts,pv,lab)
#
# This macro is to plot two screens, one is to plot the
# likelihood ratio and test of the statistical hypothesis,
# the other is to plot the power function and the result of
# the statistical test.
# ( Right tail test for unknown sigma )
#
({
  par(mfrow=c(2,1), oma=c(5,3,0,0))
  mx_mxn[1]
  mn_mxn[2]
  maxy_max(tx0, tx1)
#
# Plot the Likelihood Function
#
  plot(x3,lr, type="l", xlab="T", ylab="",
       err=-1)
  segments(tc,0,tc,kv)
  title(main="Likelihood Ratio Function", sub=lab[4])
#
# Plot the T distribution under the null and the noncentral T
#
  plot(x3,pt0, xlab="T", ylab="", type="l",
       xlim=c(mn,mx), ylim=c(0, maxy), err=-1)
  lines(x4, pt1)
  title(main="Test of Statistical Hypothesis",
       sub="Ho : Left    Ha : Right")
  mtext(side=2, line=1, outer=T,
    "    Lambda            PDF      ")
  segments(0,tx0,0,0)
  segments(ncp,tx1,ncp,0)
#
# Plot the T-critical value
#
  abline(v=tc)
#
# Find the alpha area
#
  al_x3[x3 > tc]
  nal_len(al)
  al_c(tc, al)  # X coordinates for alpha area
  ay_pt0[(101-nal):100]
  ay_c(tc0, ay)  # Y coordinates for alpha area
#
# Find the beta area
#
  be_x4[x4 < tc]
  nbe_len(be)
  be_c(be, tc)   # X coordinates for beta area
  by_pt1[1:(max(nbe,1))]
  by_c(by, tc1)   # Y coordinates for beta area
```

```
#
# Shade the beta area
#
  if(nbe>0) {
  hatch(c(min(x4),be,tc), c(0,by,0), space=.05, angle=135, border=F)
  }
#
# Shade the alpha area
#
  hatch(c(tc,al,max(x3)), c(0,ay,0), space=.05, border=F)
  mtext(side=1, line=1, outer=T, lab[3])
  mtext(side=1, line=3, outer=T, lab[1])
  mtext(side=1, line=4, outer=T, lab[2])
#
# Third screen
#
# Plot the Power Function
#
  plot(pwrmu,pwr, xlab="MU", ylab="", type="l", err=-1)
  segments(mu1,pw1,mu1,0)
  title(main="Power Function", sub=lab[6])
#
# Plot the T distribution under the null and the noncentral T
#
  plot(x3,pt0, xlab="T", ylab="", type="l",
       xlim=c(mn,mx), ylim=c(0, maxy),  err=-1)
  lines(x4, pt1)
#
# Plot the T-critical value
#
  abline(v=tc)
#
#
# Shade the beta area
#
  if(nbe>0) {
  hatch(c(min(x4),be,tc), c(0,by,0), space=.05, angle=135, border=F)
  }
#
# Shade the alpha area
#
  hatch(c(tc,al,max(x3)), c(0,ay,0), space=.05, border=F)
#
# Plot the T * value
#
  arrows(ts, tx0/2, ts, 0)
  title(main="Result of Statistical Hypothesis Test",
       sub=lab[3])
  mtext(side=2, line=1, outer=T,
    "    POWER              PDF      ")
  mtext(side=1, line=1, outer=T, lab[1])
  mtext(side=1, line=2, outer=T, lab[2])
```

```
  mtext(side=1, line=3, outer=T, lab[5])
  dcs_ifelse(ts>tc, "Reject Ho", "Do Not Reject Ho")
  mtext(side=1, line=4, outer=T,
        encode("Decision : ",dcs,"   P-value =",pv))
  rm(mx,mn,maxy,al,nal,ay,be,nbe,by,dcs)
})
END
```

```
MACRO screen23lp(ncp,tc,tc0,tc1,x3,x4,pt0,pt1,mxn,lr,kv,
               tx0,tx1,pwrmu,pwr,pw1,mu1,ts,pv,lab)
#
# This macro is to plot two screens, one is to plot the
# likelihood ratio and test of the statistical hypothesis,
# the other is to plot the power function and the result of
# the statistical test.
# ( Left tail test for unknown sigma )
#
({
  mx_mxn[1]
  mn_mxn[2]
  par(mfrow=c(2,1), oma=c(5,3,0,0))
#
# Plot the Likelihood Function
#
  plot(x3,lr, type="l", xlab="T", ylab="",
       err=-1)
  segments(tc,0,tc,kv)
  title(main="Likelihood Ratio Function", sub=lab[4])
#
# Plot the T distribution under the null and the noncentral T
#
  plot(x3,pt0, xlab="T", ylab="", type="l",
       xlim=c(mn,mx), err=-1)
  lines(x4, pt1)
  title(main="Test of Statistical Hypothesis",
       sub="Ho : Right    Ha : Left")
  mtext(side=2, line=1, outer=T,
    "    Lambda              PDF      ")
  segments(0,tx0,0,0)
  segments(ncp,tx1,ncp,0)
#
# Plot the T-critical value
#
  abline(v=tc)
#
# Find the alpha area
#
  al_x3[x3 < tc]
  nal_len(al)
  al_c(al, tc)  # X coordinates for alpha area
  ay_pt0[1:nal]
  ay_c(ay, tc0) # Y coordinates for alpha area
```

```
#
# Find the beta area
#
   be_x4[x4 > tc]
   nbe_len(be)
   be_c(tc, be)   # X coordinates for beta area
   by_pt1[(min(64, (65-nbe))) : 64]
   by_c(tc1, by)  # Y coordinates for beta area
#
# Shade the beta area
#
   if(nbe>0) {
   hatch(c(tc,be,max(x4)), c(0,by,0), space=.05, angle=135, border=F)
   }
#
# Shade the alpha area
#
   hatch(c(min(x3),al,tc), c(0,ay,0), space=.05, border=F)
   mtext(side=1, line=1, outer=T, lab[3])
   mtext(side=1, line=3, outer=T, lab[1])
   mtext(side=1, line=4, outer=T, lab[2])
#
# Third screen
#
# Plot the Power Function
#
   plot(pwrmu,pwr, xlab="MU", ylab="", type="l", err=-1)
   segments(mu1,pw1,mu1,0)
   title(main="Power Function", sub=lab[6])
#
# Plot the T distribution under the null and the noncentral T
#
   plot(x3,pt0, xlab="T", ylab="", type="l",
        xlim=c(mn,mx), err=-1)
   lines(x4, pt1)
#
# Plot the T-critical value
#
   abline(v=tc)
#
# Shade the beta area
#
   if(nbe>0) {
   hatch(c(tc,be,max(x4)), c(0,by,0), space=.05, angle=135, border=F)
   }
#
# Shade the alpha area
#
   hatch(c(min(x3),al,tc), c(0,ay,0), space=.05, border=F)
```

```
#
# Plot the T * value
#
  arrows(ts, tx0/2, ts, 0)
  title(main="Result of Statistical Hypothesis Test",
       sub=lab[3])
  mtext(side=2, line=1, outer=T,
    "    POWER               PDF     ")
  mtext(side=1, line=1, outer=T, lab[1])
  mtext(side=1, line=2, outer=T, lab[2])
  mtext(side=1, line=3, outer=T, lab[5])
  dcs_ifelse(ts<tc, "Reject Ho", "Do Not Reject Ho")
  mtext(side=1, line=4, outer=T,
        encode("Decision : ",dcs,"   P-value =",pv))
  rm(al,nal,ay,be,nbe,by,dcs,mx,mn)
})
END
```

```
MACRO screen23tp(ncp,tc,tc0,tc1,x3,x4,pt0,pt1,mxn,
             lr,kv,tx0,tx1,pwrmu,pwr,pw1,mu1,ts,pv,lab)
#
# This macro is to plot two screens, one is to plot the
# likelihood ratio and test of the statistical hypothesis,
# the other is to plot the power function and the result of
# the statistical test.
# ( Two tail test for unknown sigma )
#
({
  par(mfrow=c(2,1), oma=c(5,3,0,0))
  mx_mxn[1]
  mn_mxn[2]
#
# Plot the Likelihood Function
#
  plot(x3,lr, type="l", xlab="T", ylab="",
       err=-1)
  segments(tc,0,tc,kv)
  title(main="Likelihood Ratio Function", sub=lab[4])
#
# Plot the T distribution under the null and the noncentral T
#
  plot(x3,pt0, xlab="T", ylab="", type="l",
       xlim=c(mn,mx), ylim=c(0, max(tx0, tx1)), err=-1)
  lines(x4, pt1)
  title(main="Test of Statistical Hypothesis",
       sub="Ho : Left    Ha : Right")
  mtext(side=2, line=1, outer=T,
    "    Lambda             PDF     ")
  segments(0,tx0,0,0)
  segments(ncp,tx1,ncp,0)
#
# Plot the T-critical value
#
  abline(v=tc)
#
# Find the alpha area
#
  ar_x3[x3 > tc[1]]
  nar_len(ar)
  apr_c(tc[1], ar)  # X coordinates for right side ofalpha area
  al_x3[x3 < tc[2]]
  nal_len(al)
  apl_c(al, tc[2])  # X coordinates for left side ofalpha area
  yr_pt0[(101-nar):100]
  ayr_c(tc0[1], yr)  # Y coordinates for right side of alpha area
  yl_pt0[1:nal]
  ayl_c(yl, tc0[2])  # Y coordinates for left side of alpha area
```

```
#
# Find the beta area
#
  be_x4[(x4 < tc[1]) & (x4 > tc[2])]
  nbe_len(be)
  if (nbe > 0) {
    minb_min(be)
    ibe_len(?which(x4<=minb))
    fbe_len(be)
    bxl_max(tc[2], min(x4))
    bxr_max(tc[1], min(x4))
    be_c(bxl, be, bxr)    # X coordinates for beta area
    by_pt1[(max(1, ibe)) : (max(1, (fbe+ibe-1)))]
    miny_min(pt1)
    byl_max(miny, tc1[2])
    byr_max(miny, tc1[1])
    by_c(byl, by, byr)    # Y coordinates for beta area
#
# Shade the beta area
#
   hatch(c(bxl, be, bxr), c(0,by,0), space=.05, angle=135, border=F)
  }
#
# Shade the alpha area
#
  hatch(c(tc[1],apr,max(x3)), c(0,ayr,0), space=.05, border=F)
  hatch(c(min(x3),apl,tc[2]), c(0,ayl,0), space=.05, border=F)
  mtext(side=1, line=1, outer=T, lab[3])
  mtext(side=1, line=3, outer=T, lab[1])
  mtext(side=1, line=4, outer=T, lab[2])
#
# This is the third screen.
#
# Plot the Power Function
#
  plot(pwrmu,pwr, xlab="MU", ylab="", type="l", err=-1)
  segments(mu1,pw1,mu1,0)
  title(main="Power Function", sub=lab[6])
#
# Plot the T distribution under the null and the noncentral T
#
  plot(x3,pt0, xlab="T", ylab="", type="l",
       xlim=c(mn,mx), ylim=c(0, max(tx0, tx1)),  err=-1)
  lines(x4, pt1)
  segments(0, tx0, 0, 0)
  segments(ncp, tx1, ncp, 0)
#
# Plot the T-critical value
#
  abline(v=tc)
```

```
#
# Shade the beta area
#
   if (nbe > 0) {
    hatch(c(bxl, be, bxr), c(0,by,0), space=.05, angle=135, border=F)
    rm(minb, ibe, fbe, bxl, bxr, miny, byl, byr, by)
   }
#
# Shade the alpha area
#
  hatch(c(tc[1],apr,max(x3)), c(0,ayr,0), space=.05, border=F)
  hatch(c(min(x3),apl,tc[2]), c(0,ayl,0), space=.05, border=F)
#
# Plot the T * value
#
  arrows(ts, tx0/2, ts, 0)
  title(main="Result of Statistical Hypothesis Test",
       sub=lab[3])
  mtext(side=2, line=1, outer=T,
    "    POWER              PDF     ")
  mtext(side=1, line=1, outer=T, lab[1])
  mtext(side=1, line=2, outer=T, lab[2])
  mtext(side=1, line=3, outer=T, lab[5])
  dcs_ifelse(abs(ts) > abs(tc[1]), "Reject Ho", "Do Not Reject Ho")
  mtext(side=1, line=4, outer=T,
        encode("Decision : ",dcs,"   P-value =",pv))
  rm(mx,mn,al,ar,apl,apr,nal,nar,yr,yl,ayr,ayl,be,nbe,dcs)
})
END
```

```
MACRO power
({
   item_c("Get the vaule of power",
          "Get the power function curve")
   action_c("?pwrone",
            "?pwrtwo")
   menu( item, action)
   rm( item, action)
})
END



MACRO pwrone
({
  item_c("Right tail test",
         "Left  tail test",
         "Two   tail test")
   action_c("?pwr1",
            "?pwr2",
            "?pwr3")
   menu( item, action)
   rm( item, action)
})
END



MACRO pwr1(
m0/?PROMPT(MUo          : )/,
m1/?PROMPT(MU1          : )/,
sg/?PROMPT(Sigma        : )/,
nn/?PROMPT(Sample Size  : )/,
al/?PROMPT(Alpha        : )/)
chapter
({
   ?T(m0)_m0
   ?T(m1)_m1
   ?T(s)_sg
   ?T(n)_nn
   ?T(a)_al
   ?T(ss)_?T(s)/sqrt(?T(n))
   ncp_(?T(m1)-?T(m0))/?T(ss)
   dof_?T(n)-1
   qtl_qt((1-?T(a)), dof)
   ?T_1-nctp(ncp, dof, qtl)
   rm(?T(m0), ?T(m1), ?T(s), ?T(n), ?T(a), ?T(ss),
      ncp, dof, qtl, ?T, value=?T)
})
END
```

```
MACRO pwr2(
m0/?PROMPT(MUo            : )/,
m1/?PROMPT(MU1            : )/,
sg/?PROMPT(Sigma          : )/,
nn/?PROMPT(Sample Size    : )/,
al/?PROMPT(Alpha          : )/)
chapter
({
   ?T(m0)_m0
   ?T(m1)_m1
   ?T(s)_sg
   ?T(n)_nn
   ?T(a)_al
   ?T(ss)_?T(s)/sqrt(?T(n))
   ncp_(?T(m1)-?T(m0))/?T(ss)
   dof_?T(n)-1
   qtl_qt(?T(a), dof)
   ?T_nctp(ncp, dof, qtl)
   rm(?T(m0),?T(m1),?T(s),?T(n),?T(a),?T(ss),ncp,dof,
      qtl, ?T, value=?T)
})
END
```

```
MACRO pwr3(
m0/?PROMPT(MUo            : )/,
m1/?PROMPT(MU1            : )/,
sg/?PROMPT(Sigma          : )/,
nn/?PROMPT(Sample Size    : )/,
al/?PROMPT(Alpha          : )/)
chapter
({
   ?T(m0)_m0
   ?T(m1)_m1
   ?T(s)_sg
   ?T(n)_nn
   ?T(a)_al
   ?T(ss)_?T(s)/sqrt(?T(n))
   ncp_(?T(m1)-?T(m0))/?T(ss)
   dof_?T(n)-1
   qtl_qt(c((1-(?T(a)/2)),?T(a)/2), dof)
   cdfr_nctp(ncp, dof, qtl[1])
   cdfl_nctp(ncp, dof, qtl[2])
   ?T_1-cdfr+cdfl
   rm(?T(m0), ?T(m1), ?T(s), ?T(n), ?T(a), ?T(ss),
      ncp, dof, qtl, cdfr, cdfl, ?T, value=?T)
})
END
```

```
MACRO pwrtwo
({
  item_c("Right tail test",
         "Left  tail test",
         "Two   tail test")
   action_c("?pwra",
            "?pwrb",
            "?pwrc")
   menu( item, action)
   rm( item, action)
})
END
```

```
MACRO pwra(
m0/?PROMPT(MUo           : )/,
sg/?PROMPT(Sigma         : )/,
nn/?PROMPT(Sample Size   : )/,
al/?PROMPT(Alpha         : )/,
mn/?PROMPT(Min MU1 for Plot  : )/,
mx/?PROMPT(Max MU1 for Plot  : )/)
chapter
({
   par(mfrow=c(1,1), oma=c(4,2,0,0))
   ?T(m0)_m0
   ?T(s)_sg
   ?T(n)_nn
   ?T(a)_al
   ?T(m1)_seq(mn, mx, len=51)
   ?T(ss)_?T(s)/sqrt(?T(n))
   ncp_(?T(m1)-?T(m0))/?T(ss)
   dof_?T(n)-1
   qtl_qt((1-?T(a)), dof)
   ?T(y)_1-nctp(ncp, dof, qtl)
   plot(?T(m1), ?T(y), xlab="MU", ylab="",
         type="l", err=-1)
   title(main="Power Function for Right Tail Test")
   mtext(side=1, line=1, outer=T,
     encode("MUo :",?T(m0),"  Alpha :",?T(a)))
   mtext(side=1, line=3, outer=T,
     encode("Sample size :",?T(n),"  Sigma :",?T(s)))
   rm(?T(m0), ?T(m1), ?T(s), ?T(n), ?T(a), ?T(ss),
      ncp, dof, qtl, ?T(y))
})
END
```

```
MACRO pwrb(
m0/?PROMPT(MUo          : )/,
sg/?PROMPT(Sigma        : )/,
nn/?PROMPT(Sample Size  : )/,
al/?PROMPT(Alpha        : )/,
mn/?PROMPT(Min MU1 for Plot  : )/,
mx/?PROMPT(Max MU1 for Plot  : )/)
chapter
({
   par(mfrow=c(1,1), oma=c(4,2,0,0))
   ?T(m0)_m0
   ?T(s)_sg
   ?T(n)_nn
   ?T(a)_al
   ?T(m1)_seq(mn, mx, len=51)
   ?T(ss)_?T(s)/sqrt(?T(n))
   ncp_(?T(m1)-?T(m0))/?T(ss)
   dof_?T(n)-1
   qtl_qt(?T(a), dof)
   ?T(y)_nctp(ncp, dof, qtl)
   plot(?T(m1), ?T(y), xlab="MU", ylab="",
         type="l", err=-1)
   title(main="Power Function for Left tail Test")
   mtext(side=1, line=1, outer=T,
     encode("MUo :",?T(m0),"  Alpha :",?T(a)))
   mtext(side=1, line=3, outer=T,
     encode("Sample size :",?T(n),"  Sigma :",?T(s)))
   rm(?T(m0), ?T(m1), ?T(s), ?T(n), ?T(a), ?T(ss),
      ncp, dof, qtl, ?T(y))
})
END
```

```
MACRO pwrc(
m0/?PROMPT(MUo            : )/,
sg/?PROMPT(Sigma          : )/,
nn/?PROMPT(Sample Size    : )/,
al/?PROMPT(Alpha          : )/,
mn/?PROMPT(Min MU1 for Plot  : )/,
mx/?PROMPT(Max MU1 for Plot  : )/)
chapter
({
   par(mfrow=c(1,1), oma=c(4,2,0,0))
   ?T(m0)_m0
   ?T(s)_sg
   ?T(n)_nn
   ?T(a)_al
   ?T(m1)_seq(mn, mx, len=51)
   ?T(ss)_?T(s)/sqrt(?T(n))
   ncp_(?T(m1)-?T(m0))/?T(ss)
   dof_?T(n)-1
   qtl_qt(c(((1-?T(a))/2),?T(a)/2), dof)
   ?T(yr)_nctp(ncp, dof, qtl[1])
   ?T(yl)_nctp(ncp, dof, qtl[2])
   ?T(y)_1-?T(yr)+?T(yl)
   plot(?T(m1), ?T(y), xlab="MU", ylab="",
         type="l", err=-1)
   title(main="Power Function for Two Tail Test")
   mtext(side=1, line=1, outer=T,
     encode("MUo :",?T(m0),"  Alpha :",?T(a)))
   mtext(side=1, line=3, outer=T,
     encode("Sample size :",?T(n),"  Sigma :",?T(s)))
   rm(?T(m0), ?T(m1), ?T(s), ?T(n), ?T(a), ?T(ss),
      ncp, dof, qtl, ?T(y),?T(yl),?T(yr))
})
END
```

Appendix E: S Interface Routines and FORTRAN Programs

Table of Contents

```
FUNCTION bvnorm(
         mu1/REAL/
         sigma1/REAL/
         mu2/REAL/
         sigma2/REAL/
         rho/REAL/
         interv/INT/
         zval/REAL/
         )
 STRUCTURE(
       bvcdf/REAL, interv/
          )
    call bvnormf(mu1,sigma1,mu2,sigma2,rho, interv,zval,bvcdf)
    RETURN(bvcdf)
END
```

```
      subroutine bvnormf(mu1,sigma1,mu2,sigma2,rho,interv,zval,bvcdf)
c
c ==================================================================
c PROGRAM DESCRIPTION:
c
c  This program computes the AREA of a BIVARIATE NORMAL
c  DISTRIBUTION with means: mu1 and mu2, standard deviation: sigma1
c  and sigma 2 with corrlation rho.  It uses the IMSL routine
c  "dblin" which is designed to perform a double integration
c  of the two variable function F(X,Y).
c
c
c ==================================================================
c
      integer m,interv,ier
      real bvcdf(interv),aerr,error,dblin
c
      external f1,ay,by
c
      real xl,xh,zlow,zhigh,zval(interv+1)
      real mu1,mu2,sigma1,sigma2,rho
      real mux,muy,sigmax,sigmay,ro
      common /bounds/ zlow,zhigh
      common /param/ mux,muy,sigmax,sigmay,ro
c
c ===================================================
c
      aerr = 0.01
      mux = mu1
      muy = mu2
      sigmax = sigma1
      sigmay = sigma2
      ro = rho
      xl = mu1 - 6*sigma1
      xh = mu1 + 6*sigma1
c
c
      do 10 m = 1,interv
c
c =========================================================
c
c Here we have to set the minimum and maximum values of
c the inner integral, place the values in common, and then
c call "dblin"...We note it uses two external functions,
c "ay" and "by" which can be used to allow integration
c of limits in terms of the outer variable...Here, the
c limits are (LOWER LIMIT = zs - x)
c            (UPPER LIMIT = ze - x),
c where zs = zstart + zinc * ( i - 1 )
c       ze = zstart + zinc * i
c And the outer limit can be set up followings ;
c LOWER LIMIT = mu1 - 6 * sigma1
```

```
c UPPER LIMIT = mu1 + 6 * sigma1
c
c ==========================================================
c
          zlow  = zval(m)
          zhigh = zval(m+1)
          bvcdf(m)=dblin(f1,xl,xh,ay,by,aerr,error,ier)
10      continue
        return
        end
c
c ==================================================================
c ==================================================================
c
        real function f1(x,y)
c
c This function is used by IMSL routine dblin to compute
c the AREA under the bivariate normal distribution.
c
        real x,y,a,b,c,k1,k2
c
        real mux,muy,sigmax,sigmay,ro
        common /param/ mux,muy,sigmax,sigmay,ro
c
        real pi
        pi = 3.1415927
c
        k1 = 1/(2*pi*sigmax*sigmay*sqrt(1-ro**2))
        a  = ((x-mux)**2)/sigmax**2
        b  = 2*ro*((x-mux)/sigmax)*((y-muy)/sigmay)
        c  = ((y-muy)**2)/sigmay**2
        k2 = -(1/2.0)*(1/(1-ro**2))
        f1 = k1*exp(k2*(a-b+c))
        return
        end
c
c ==============================================================
c ==============================================================
c
        real function ay(x)
c
c This function computes the lower limit of the inner
c integral of the bivariate normal density
c
          real zlow,zhigh
          common /bounds/ zlow,zhigh
            ay = zlow - x
          return
        end
c
c ==============================================================
c ==============================================================
```

```
c
c
      real function by(x)
c
c This functions computes the upper limit of the
c  inner integral of the bivariate normal density
c
        real zlow,zhigh
        common /bounds/ zlow,zhigh
           by = zhigh - x
        return
      end
```

```
FUNCTION bvnorm2(
         mu1/REAL/
         sigma1/REAL/
         mu2/REAL/
         sigma2/REAL/
         rho/REAL/
         interv/INT/
         zval/REAL/
         )
 STRUCTURE(
       bvcdf/REAL, interv+1/
          )
    call bvnorm2f(mu1,sigma1,mu2,sigma2,rho, interv,zval,bvcdf)
    RETURN(bvcdf)
END
```

```
      subroutine bvnorm2f(mu1,sigma1,mu2,sigma2,rho,interv,zval,bvcdf)
c
c ==================================================================
c PROGRAM DESCRIPTION:
c
c  This program computes the AREA of a BIVARIATE NORMAL
c  DISTRIBUTION with means: mu1 and mu2, standard deviation: sigma1
c  and sigma 2 with corrlation rho.  It uses the IMSL routine
c  "dblin" which is designed to perform a double integration
c  of the two variable function F(X,Y).
c  Transform : Z = X^2 + Y^2
c
c
c ==================================================================
c
       integer m,interv,ier
       real bvcdf(interv+1),aerr,error,dblin
c
       external f1,ay,by
c
       real xl,xh,zval(interv+1),z
       real mu1,mu2,sigma1,sigma2,rho
       real mux,muy,sigmax,sigmay,ro
       common /bounds/ z
       common /param/ mux,muy,sigmax,sigmay,ro
c
c ===================================================
c
       aerr = 0.01
       mux = mu1
       muy = mu2
       sigmax = sigma1
       sigmay = sigma2
       ro = rho
c
c
       do 10 m = 1,(interv+1)
c
c =========================================================
c
c Here we have to set the minimum and maximum values of
c the inner integral, place the values in common, and then
c call "dblin"...We note it uses two external functions,
c "ay" and "by" which can be used to allow integration
c of limits in terms of the outer variable...Here, the
c limits are (LOWER LIMIT = sqrt(z - x^2)
c            (UPPER LIMIT = sqrt(z - x^2),
c And the outer limit can be set up followings ;
c LOWER LIMIT = sqrt(z)
c UPPER LIMIT = -sqrt(z)
c
c This function will return the cdf for the circle
```

```
c whose radius is sqrt(z). Therefore, to get the volumn
c under the ring shape, the previous voulmun should be
c subtracted. It will be done in S program.
c
c ============================================================
c
          z  = zval(m)
          xh  = sqrt(z)
          xl  = -xh
          bvcdf(m) = dblin(f1,xl,xh,ay,by,aerr,error,ier)
10      continue
        return
        end
c
c ====================================================================
c ====================================================================
c
        real function f1(x,y)
c
c This function is used by IMSL routine dblin to compute
c the AREA under the bivariate normal distribution.
c
        real x,y,a,b,c,k1,k2
c
        real mux,muy,sigmax,sigmay,ro
        common /param/ mux,muy,sigmax,sigmay,ro
c
        real pi
        pi = 3.1415927
c
        k1 = 1/(2*pi*sigmax*sigmay*sqrt(1-ro**2))
        a  = ((x-mux)**2)/sigmax**2
        b  = 2*ro*((x-mux)/sigmax)*((y-muy)/sigmay)
        c  = ((y-muy)**2)/sigmay**2
        k2 = -(1/2.0)*(1/(1-ro**2))
        f1 = k1*exp(k2*(a-b+c))
        return
        end
c
c ================================================================
c ================================================================
c
        real function ay(x)
c
c This function computes the lower limit of the inner
c integral of the bivariate normal density
c
          real z,x
          common /bounds/ z
            ay = -sqrt(z - x**2)
          return
        end
```

```
c
c ==============================================================
c ==============================================================
c
c
      real function by(x)
c
c This functions computes the upper limit of the
c inner integral of the bivariate normal density
c
        real z,x
        common /bounds/ z
           by = sqrt(z - x**2)
        return
      end
```

```
FUNCTION bvunif(
          interv/INT/
          zval/REAL/
          )
 STRUCTURE(
       bvcdf/REAL, interv/
           )
    call bvuniff( interv,zval,bvcdf)
    RETURN(bvcdf)
END
```

```
      subroutine bvuniff( interv,zval,bvcdf)
c
c ===================================================================
c PROGRAM DESCRIPTION:
c
c  This program computes the AREA of a BIVARIATE UNIFORM
c  DISTRIBUTION which distributes from 0 to 1, and the transform
c  is z = x + y.
c  It uses the IMSL routine "dblin" which is designed to
c  perform a double integration of the two variable function F(X,Y).
c
c ===================================================================
c
       integer m,interv,ier
      real bvcdf( interv),zval( interv+1),cdtot,cdsub
      real aerr,error,dblin
c
      external f1,ay1,ay2,by1,by2
c
      real z,xhi,xh
      common /bounds/ z
c
c ==================================================
c
      aerr = 0.001
c
c
      do 10 m = 1,interv
c
c ========================================================
c
c Here we have to set the minimum and maximum values of
c the inner integral, which is defined ay1, ay2, and by1,by2 and then
c call "dblin"...We note it uses four external functions,
c "ay1", "ay2", and "by1" "by2" which can be used to allow intergration
c of limits in terms of the outer variable...Here, the
c ay1 : lower limit for total area ( = 0 )
c ay2 : lower limit for subtracting part ( = 1 )
c by1 : upper limit for total area ( = z - x )
c by2 : upper limit for subtracting part
c       by2 = 1 if z <= 1  or  by2 = z-x  if  z > 1
c
c To compute the cdf which distributed uniform(0,1),
c first the area  under the y=z-x is computed,
c then, if the z > 1, the area  of out side of the square
c is computed and subtracted from the area under y=z-x.
c
c ========================================================
c
        z = zval(m+1)
        xhi  = min ( 1, z )
        xh   = max ( 0, z-1 )
```

```
          cdtot = dblin(f1,0,xhi,ay1,by1,aerr,error,ier)
          cdsub = dblin(f1,0,xh ,ay2,by2,aerr,error,ier)
          bvcdf(m)= cdtot - cdsub
10      continue
        return
        end
c
c ==================================================================
c ==================================================================
c
        real function f1(x,y)
c
c This function is used by IMSL routine dblin to compute
c the AREA under the bivariate uniform distribution.
c
c
        f1 = 1.0
        return
        end
c
c ==============================================================
c ==============================================================
c
        real function ay1(x)
c
c This function computes the lower limit of the inner
c integral of area under y=z-x.
c
            ay1 = 0
          return
        end
c
c ==============================================================
c ==============================================================
c
        real function ay2(x)
c
c This function computes the lower limit of the inner
c integral of the out side of the area under y=z-x
c
            ay2 = 1
          return
        end
c
c
c ==============================================================
c ==============================================================
c
c
c
        real function by1(x)
c
```

```
c This functions computes the upper limit of the
c  inner  integral of the area under y=z-x
c
          real z,x
          common /bounds/ z
             by1 =  z - x
          return
        end
c
c ==============================================================
c ==============================================================
c
c
c
        real  function by2(x)
c
c This functions computes the upper limit of the
c  inner  integral of out side of the area under y=z-x
c
          real z,x
          common /bounds/ z
           if (z .LE. 1) then
             by2 =  1
          else
             by2 = z-x
          end if
          return
        end
```

```
FUNCTION bvunif2(
          interv/INT/
          zval/REAL/
          )
 STRUCTURE(
       bvcdf/REAL,interv/
           )
    call bvunif2f( interv,zval,bvcdf)
    RETURN(bvcdf)
END
```

```
      subroutine bvunif2f( interv,zval,bvcdf)
c
c =================================================================
c PROGRAM DESCRIPTION:
c
c  This program computes the AREA of a BIVARIATE UNIFORM
c  DISTRIBUTION which distributes from 0 to 1, and the transform
c  is z = x**2 + y**2.
c  It uses the IMSL routine "dblin" which is designed to
c  perform a double integration of the two variable function F(X,Y).
c
c =================================================================
c
      integer m,interv,ier
      real bvcdf( interv),zval( interv+1),cdtot,cdsub
      real aerr,error,dblin
c
      external f1,ay1,ay2,by1,by2
c
      real z,xhi,xh
      common /bounds/ z
c
c ================================================
c
      aerr = 0.001
c
c
      do 10 m = 1,interv
c
c ==========================================================
c
c Here we have to set the minimum and maximum values of
c the inner integral, which is defined ay1, ay2, and by1, and then
c call "dblin"...We note it uses three external functions,
c "ay1", "ay2", and "by1" which can be used to allow intergration
c of limits in terms of the outer variable...Here, the
c ay1 : lower limit for total quarter citcle ( = 0 )
c ay2 : lower limit for subtracting part ( = 1 )
c by1 : upper limit for  both total quarter citcle and
c subtracting part ( sqrt(z-x**2) )
c
c To compute the cdf which distributed uniform(0,1),
c first the area  of quarter circle whose radius is sqrt(z),
c is computed, then, if the sqrt(z) is greater than 1,
c the area  of out side of the square is computed and
c subtracted from the area of the quarter circle.
c
c ==========================================================
c
        z = zval(m+1)
        xhi   = min ( 1, sqrt(z) )
        xh    = sqrt( max(1,z) - 1 )
```

```fortran
          cdtot = dblin(f1,0,xhi,ay1,by1,aerr,error,ier)
          cdsub = dblin(f1,0,xh ,ay2,by2,aerr,error,ier)
          bvcdf(m)= cdtot - cdsub
10      continue
        return
        end
c
c ====================================================================
c ====================================================================
c
        real function f1(x,y)
c
c This function is used by IMSL routine dblin to compute
c the AREA under the bivariate normal distribution.
c
c
        f1 = 1.0
        return
        end
c
c ================================================================
c ================================================================
c
        real function ay1(x)
c
c This function computes the lower limit of the inner
c integral of quarter circle.
c
            ay1 = 0
          return
        end
c
c ===============================================================
c ===============================================================
c
        real function ay2(x)
c
c This function computes the lower limit of the inner
c integral of the out side of the quarter circle.
c
            ay2 = 1
          return
        end
c
c
c ===============================================================
c ===============================================================
c
c
c
        real function by1(x)
c
```

```
c This functions computes the upper limit of the
c inner integral of the quarter circle.
c
          real z,x
          common /bounds/ z
             by1 = sqrt ( z - x**2 )
          return
        end
c
c ==============================================================
c ==============================================================
c
c
c
        real function by2(x)
c
c This functions computes the upper limit of the
c inner integral of out side of the quarter circle.
c
          real z,x
          common /bounds/ z
             by2 = max( 1, sqrt (z - x**2) )
          return
        end
```

```
FUNCTION bvchisq(
         df1/REAL/
         df2/REAL/
         interv/INT/
         gam1/REAL/
         gam2/REAL/
         yhi/REAL/
         zval/REAL/
         )
 STRUCTURE(
       bvcdf/REAL, interv/
          )
    call bvchisqf(df1,df2, interv,gam1,gam2,yhi,zval,bvcdf)
    RETURN(bvcdf)
END
```

```
      subroutine bvchisqf(df1,df2,interv,gam1,gam2,yhi,zval,bvcdf)
c
c ==============================================================
c PROGRAM DESCRIPTION:
c
c  This program computes the AREA of a BIVARIATE CHISQUARE
c  DISTRIBUTION with degree of freedom : df1 and df2.
c  It uses the IMSL routine "dblin" which is designed to
c  perform a double integration of the two variable function F(X,Y).
c  The transform is : Z = (X/DFx) / (Y/DFy).
c
c ==============================================================
c
       integer m,interv,ier
      real bvcdf(interv),aerr,error,dblin
c
      external f1,ax,bx
c
      real yhi,zlow,zhigh,zval(interv+1)
      real df1,df2,dfx,dfy,gam1,gam2,gamx,gamy,dfr
      common /bounds/ zlow,zhigh,dfr
      common /param/  dfx,dfy,gamx,gamy
c
c ==============================================
c
       aerr = 0.01
      dfx = df1
      dfy = df2
      dfr = dfx/dfy
      gamx = gam1
      gamy = gam2
c
c
       do 10 m = 1,interv
c
c ====================================================
c
c Here we have to set the minimum and maximum values of
c the inner integral, place the values in common, and then
c call "dblin"...We note it uses two external functions,
c "ax" and "bx" which can be used to allow integration
c of limits in terms of the outer variable...Here, the
c limits are (LOWER LIMIT = zs * y * DFx/DFy)
c            (UPPER LIMIT = ze * y * DFx/DFy)
c where zs = zstart + zinc * ( i - 1 )
c       ze = zstart + zinc * i
c And the outer limit can be set up followings ;
c LOWER LIMIT = 0
c UPPER LIMIT = qchisq(.9999,dfy)
c
c ====================================================
c
```

```
         zlow  = zval(m)
         zhigh = zval(m+1)
         bvcdf(m)=dblin(f1,0,yhi,ax,bx,aerr,error,ier)
10      continue
        return
        end
c
c ====================================================================
c ====================================================================
c
        real function f1(y,x)
c
c This function is used by IMSL routine dblin to compute
c the AREA under the bivariate chisquare distribution.
c
        real x,y,a1,a2,b1,b2,k1,k2
c
        real dfx,dfy,gamx,gamy
        common /param/ dfx,dfy,gamx,gamy
c
        k1 = 1/((2**(dfx/2))*gamx)
        k2 = 1/((2**(dfy/2))*gamy)
        a1 = x**((dfx/2)-1)
        a2 = y**((dfy/2)-1)
        b1 = exp(-x/2)
        b2 = exp(-y/2)
        f1 = k1*k2*a1*a2*b1*b2
        return
        end
c
c ================================================================
c ================================================================
c
        real function ax(y)
c
c This function computes the lower limit of the inner
c integral of the bivariate chisquare density
c
          real zlow,zhigh,y,dfr
          common /bounds/ zlow,zhigh,dfr
            ax = zlow * y * dfr
          return
        end
c
c ================================================================
c ================================================================
c
c
        real function bx(y)
c
c This functions computes the upper limit of the
c inner integral of the bivariate chisquare density
```

```
c
      real zlow,zhigh,y,dfr
      common /bounds/ zlow,zhigh,dfr
         bx = zhigh * y * dfr
      return
    end
```

```
FUNCTION bvnandc(
          df/REAL/
          interv/INT/
          gam/REAL/
          yhi/REAL/
          zval/REAL/
          )
 STRUCTURE(
        bvcdf/REAL, interv/
           )
    call bvncf(df, interv,gam,yhi,zval,bvcdf)
    RETURN(bvcdf)
END
```

```
      subroutine bvncf(df,interv,gam,yhi,zval,bvcdf)
c
c ===================================================================
c PROGRAM DESCRIPTION:
c
c  This program computes the AREA of a BIVARIATE of Normal (0,1)
c  and Chisquare with DF.
c  It uses the IMSL routine "dblin" which is designed to
c  perform a double integration of the two variable function F(X,Y).
c  The transform is : Z = X / Y,
c        where, X : Normal variable  N (0,1)
c               Y : Sqrt( Y' / n )
c                   Y' : Chisquare variable with n DF
c
c ===================================================================
c
      integer m,interv,ier
      real bvcdf( interv),aerr,error,dblin
c
      external f1,ax,bx
c
      real yhi,zlow,zhigh,zval( interv+1)
      real df,dfy,gam,gamy
      common /bounds/ zlow,zhigh
      common /param/  dfy,gamy
c
c ==================================================
c
      aerr = 0.001
      dfy = df
      gamy = gam
c
c
      do 10 m = 1,interv
c
c ==========================================================
c
c Here we have to set the minimum and maximum values of
c the inner integral, place the values in common, and then
c call "dblin"...We note it uses two external functions,
c "ax" and "bx" which can be used to allow integration
c of limits in terms of the outer variable...Here, the
c limits are LOWER LIMIT = zs * SQRT( y / DFy)
c            UPPER LIMIT = ze * SQRT( y / DFy)
c where zs = zstart + zinc * ( m - 1 )
c       ze = zstart + zinc * m
c And the outer limit can be set up followings ;
c LOWER LIMIT = 0
c UPPER LIMIT = qchisq(.9999,dfy)
c
c ==========================================================
c
```

```
          zlow  = zval(m)
          zhigh = zval(m+1)
          bvcdf(m)=dblin( f1,0,yhi,ax,bx,aerr,error, ier)
10      continue
        return
        end
c
c ==================================================================
c ==================================================================
c
        real function f1(y,x)
c
c This function is used by IMSL routine dblin to compute
c the AREA under the bivariate Normal and  Chisquare variable.
c
        real x,y,a1,a2,b1,b2,b3
c
        real dfy,gamy
        common /param/ dfy,gamy
c
        a1 = 1/sqrt(2*3.1415927)
        a2 = exp(-(x**2)/2)
        b1 = 1/((2**(dfy/2))*gamy)
        b2 = y**((dfy/2)-1)
        b3 = exp(-y/2)
        f1 = a1*a2*b1*b2*b3
        return
        end
c
c ==============================================================
c ==============================================================
c
        real function ax(y)
c
c This function computes the lower limit of the inner integral.
c
          real zlow,zhigh,y,dfy,gamy
          common /bounds/ zlow,zhigh
          common /param/  dfy,gamy
            ax = zlow * sqrt( y / dfy )
          return
        end
c
c ===============================================================
c ===============================================================
c
c
        real function bx(y)
c
c This functions computes the upper limit of the inner integral.
c
          real zlow,zhigh,y,dfy,gamy
```

```
      common /bounds/ zlow,zhigh
      common /param/  dfy,gamy
         bx = zhigh * sqrt( y / dfy )
      return
    end
```

```
FUNCTION nct(
      ncp /REAL/
       df /REAL/
   splits /REAL/
       )
STATIC( integer leng)
   leng = 2**splits
STRUCTURE (
       ord /REAL,leng/
   density /REAL,leng/
       qtl /REAL,leng/
       cdf /REAL,leng/
      )
   call nctden(ncp,df,splits,ord,density,qtl,cdf,LENGTH(ord))
RETURN (ord,density,qtl,cdf)
END
```

```
      subroutine nctden(ncp,df,splits,ord,density,qtl,cdf,m)
c
c ==================================================================
c  This subroutine computes the density of the noncentral
c  t distribution and the CDF of the same density
c  by computing cdf values for the endpoints of an
c  interval {t,t+dt}, and then dividing by dt to obtain
c  the approximate height of the density at (t+dt/2)
c
c  Note: One can obtain a better approximation for the
c        density by asking for more splits when calling
c        the S function: nct CAUTION: The resolution of
c        the plotting device should always be considered
c        since the total number of ordinal values: 2^splits + 1
c        should not exceed the number of character positions
c        of the plotting device
c
c  Note: The NONCENTRALITY PARAMETER should be computed
c        before calling the S function "nct" ... it is
c        equal to: (mu1-mu0)/(s/sqrt(n))
c
c ==================================================================
      integer ier,nord,degf,m
      real ncp,df,splits,ord(m),density(m),tprob1,tprob2
      real qtl(m),cdf(m)
c
c   Compute the number of lower bound ordinate values
c
      degf= df
      nord = 2**splits
c
c   Compute the length of each interval between equally
c    spaced ordinate values
c
      dt = ((ncp+4) - (ncp-4))/nord
c
c   Set the lower and upper bound of the first interval
c
      t1 = ncp - 4
      t2 = t1 + dt
c
c   Compute the CDF values of each boundary value and
c    compute the height of the density.  Store this value and
c    the cumulative probability of the upper bound of
c    the interval under consideration
c
      do 10 j = 1,nord
c
        call mdtn(t1,degf,ncp,tprob1,ier)
        call mdtn(t2,degf,ncp,tprob2,ier)
c
c    Set the values for the approximate density
```

```
c      ord(j) is the ordinal value
c      density(j) is the height of the density at ord(j)
c
          ord(j) = (t1+t2)/2
          density(j) = (tprob2-tprob1)/dt
c
c    Set the values for the CDF of the Noncentral t
c      qtl(j) is the quantile value
c      cdf(j) is the Probability of t <= qtl(j)
c
          qtl(j) = t2
          cdf(j) = tprob2
c
c   Set the lower and upper bound of the next interval
c    for which computation is required
c
          t1 = t2
          t2 = (ncp-4) + (j+1)*dt
c
10        continue
          return
          end
```

```
FUNCTION nctp(
      ncp /REAL/
       df /REAL/
      qtl /REAL/
      )
STRUCTURE (
       cdf /LIKE(ncp)/
      )
   call nctsub(ncp,df,qtl,cdf,LENGTH(ncp))
RETURN (cdf)
END
```

```
        subroutine nctsub(ncp,df,qtl,cdf,m)
c
c ==================================================================
c  This subroutine computes the cdf of the noncentral
c  t distribution.
c
c  Note: The NONCENTRALITY PARAMETER should be computed
c        before calling the S function "ncst" ... it is
c        equal to: (mu1-mu0)/(s/sqrt(n))
c
c ==================================================================
        integer ier,degf,m
        real ncp(m),df,tprob
        real qtl,cdf(m)
        degf= df
c
c
        do 10 j = 1, m
          call mdtn(qtl,degf,ncp(j),tprob,ier)
          cdf(j) = tprob
10      continue
        return
        end
```

Bibliography

1. Banks, Jerry and John S. Carson II. Discrete-Event System Simulation. Englewood Cliffs NJ: Prentice-Hall, Inc., 1984.

2. Barr, David R. Notes on Using S for Discrete Probabilities. Lecture material distributed in MA 5.82, Theory of Probability. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1986.

3. -----. "Use of S in Statistics Courses," Proceedings on the Teaching of Statistics. Paper presented to the Annual Meeting of the American Statistical Association, Las Vegas, 1985.

4. Becker, Richard A. and John M. Chambers. Extending the S System. Monterey CA: Wadsworth Advanced Book Program, a Division of Wadsworth, Inc., 1985.

5. -----. S An Interactive Environment for Data Analysis and Graphics. Belmont CA: Wadsworth Advanced Book Program, a Division of Wadsworth, Inc., 1984.

6. Bickel, Peter J. and Kjell A. Doksum. Mathematical Statistics. Oakland CA: Holden-Day, Inc., 1977.

7. Cohen, J. Statistical Power Analysis in Behavioral Science. New York: Academic Press, 1969.

8. Davis, Gordon B. and Margrethe H. Olson. Management Information Systems. New York: McGraw-Hill Book Company, 1985.

9. Devore, Jay L. Probability and Statistics for Engineering and Sciences. Monterey CA: Brooks/Cole Publishing Company, 1982.

10. DuMouchel, William H. "Comment," The American Statistician, 33: 30-31 (February 1979).

11. IMSL Library Volume I, The. Dallas: IMSL, Inc., 1982.

12. IMSL Library Volume III, The. Dallas: IMSL, Inc., 1982.

13. Insead, Spyros Makridakis. "CLT: An Interactive Approach for Illustrating the Central Limit Theorem," The American Statistician, 33: 90 (May 1979).

14. Mahmoud, Essam and Timothy A. Davidson. "Interactive Statistical Programs: A Comprehensive Tool for Teaching Statistics," The American Statistician, 39: 140 (May 1985).

15. Maivald, Olga. "Simulation as a Teaching Method using Interactive Computer Graphics in Color," in personal correspondence to Daniel E. Reynolds, AFIT/ENC, December 1985.

16. Mendenhall, William and others. Mathematical Statistics with Applications. Boston: Duxbury Press, 1981.

17. Neter, John and others. Applied Linear Statistics Model. Homewood IL: Richard D. Irwin, Inc., 1985.

18. Thisted, Ronald A. "Teaching Statistical Computing Using Computer Packages," The American Statistician, 33: 27-35+ (February 1979).

19. Welsch, Roy E. "Comment," The American Statistician, 33: 34 (February 1979).

Vita

Captain Hyung C. Kim was born on 7 November 1957 in Daejon, Republic of Korea. After getting a GED from the Seoul Educational Board in 1975, he attended the Air Force Academy, Republic of Korea, from which he received the degree of Bachelor of Science in Aeronautical Engineering in April 1980. Upon graduation, he received a commission in the ROKAF. He completed combat pilot training and received his wings in April 1981. He then served as a F-5 pilot in the 201st Tactical Combat Squadron, Suwon AFB, Korea. In May 1983, he was assigned as a T-33 instructor pilot at the 216th Flying Training Squadron, Sachon AFB, Korea. In December 1983, he was selected to serve as an aide-de-camp to the Chief of Staff, ROK/US Combined Forces Command / Deputy Commander, USFK, YongSan, Korea until he entered the School of Systems and Logistics, Air Force Institute of Technology, in 1985.

Permanent address: 273-1 Yonhee 1 Dong
Sudaemoon-Ku, Seoul
Republic of Korea

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release; distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
AFIT/GSM/ENC/86S-10

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
School of Systems and Logistics

6b. OFFICE SYMBOL *(If applicable)*
AFIT/LSY

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State and ZIP Code)*
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433-6583

7b. ADDRESS *(City, State and ZIP Code)*

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS *(City, State and ZIP Code)*

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE *(Include Security Classification)*
See Block 19

12. PERSONAL AUTHOR(S)
Kim Hyung Chul, B.S., Captain, Republic of Korea Air Force

13a. TYPE OF REPORT
MS Thesis

13b. TIME COVERED
FROM ________ TO ________

14. DATE OF REPORT *(Yr., Mo., Day)*
1986 September

15. PAGE COUNT
466

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.
12	01	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Statistical Analysis, Statistical Distributions, Statistical Functions, Statistical Inference, Statistical Tests, Computer Graphics

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Title: DEVELOPMENT OF THE INTERACTIVE STATISTICAL TUTORIAL PACKAGE (ISTP) FOR LEARNING MATHEMATICAL CONCEPTS OF PROBABILITY AND STATISTICS

Thesis Chairman: Daniel E. Reynolds, GM-13, USAF
Assistant Professor of Computing Sciences

Approved for public release: IAW AFR 190-1.
Lynn E. Wolaver 29 Sept 86
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AFC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐

21. ABSTRACT SECURITY CLASSIFICATION

22a. NAME OF RESPONSIBLE INDIVIDUAL
Daniel E. Reynolds

22b. TELEPHONE NUMBER *(Include Area Code)*
513-255-4185

22c. OFFICE SYMBOL
AFIT/ENC

This research demonstrated the feasibility of an alternative pedagogy offered by the ISTP: one that emphasizes visual presentation and mastery of statistical concepts in lieu of the traditional theorem-proof approach to learning statistics.

Four statistical topics were selected for the initial implementation of the ISTP: probability distributions, maximum likelihood estimation, transformation of random variables and hypothesis testing. These areas were chosen because they are the main concern of a first course in statistics.

The ISTP was designed to provide an intuitive pedagogy rather than to present statistics in a cook-book fashion. The user is encouraged to think while using the ISTP and to relate the results of the ISTP to concepts of statistics.

The ISTP is hierarchically constructed so that the beginner can use this package by taking advantage of menu options. For the experienced user, it provides direct macro selection. Furthermore, the more sophisticated user can develop his own library of macros to add to those which already exist in the ISTP.

Further research should attempt to enhance current subject matter and take advantage of graphics facilities such as color graphics, animation, and windowing. More specifically, research needs to be conducted on subjects covered by current topic areas as well as more advanced subjects such as axiomatic probability, Bayesian statistics, and multivariate probability distributions. Finally, consideration should be given to porting the entire ISTP package to super computers to cope with the computational needs of forthcoming graphic packages.

END

12-86

DTIC